Servicio Semántico de Información Musical

Autor: Rocío Martínez Vidaurrázaga
Tutor: César de Pablo Sánchez

19 de Julio de 2010

Título: Servicio Semántico de Información Musical

Autor: Rocío Martínez Vidaurrázaga

Director: César de Pablo Sánchez

EL TRIBUNAL

Presidente: _____

Vocal: _____

Secretario: _____

Realizado el acto de defensa y lectura del Proyecto Fin de Carrera el día __ de _____ de 20__ en Leganés, en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, acuerda otorgarle la CALIFICACIÓN de

VOCAL

SECRETARIO PRESIDENTE

Agradecimientos

A mi tutor César de Pablo, por prestarme su ayuda en todo momento.

A todos los profesores y compañeros que durante mis años de Universidad me han ayudado a formarme tanto en lo personal como en lo profesional.

A mi familia y a mi novio por prestarme su apoyo en todo momento y por estar ahí en los malos momentos ayudándome a levantarme.

En especial me gustaría agradecer a mi madre, que tanto me ayudó durante toda mi vida y más aún en los años universitarios, animándome a seguir ante cualquier adversidad y apoyándome en todas mis decisiones.

Resumen

El presente proyecto de fin de carrera trata de construir una aplicación Web que sea capaz de realizar el análisis semántico de un texto relacionado con el mundo de la música y de enriquecer este análisis con información musical adyacente.

El sistema resultante va a integrar la información prestada por dos servicios Web externos para conseguir este propósito.

El análisis semántico va a ser realizado con la ayuda del servicio Web semántico OpenCalais, que recibirá el texto y devolverá las entidades reconocidas según el contexto. Esta respuesta será tratada por el sistema para escoger las pertenecientes al ámbito musical y con ellas realizar una petición de información a la base de datos de MusicBrainz, accediendo a la misma a través de su servicio Web. El sistema trabajará esta información para presentarla de manera amigable y atractiva al usuario final.

Palabras clave:

- Servicios Web
- Web semántica
- OpenCalais
- MusicBrainz

Abstract

This dissertation aims to build a Web application that is capable of performing a semantic analysis of text associated with the world of music and enriches this analysis with adjacent musical information.

The resulting system will integrate the information provided by two external Web services to achieve this purpose.

The semantic analysis will be done with the help of semantic Web service OpenCalais that will receive the text and will return the entities depending on context. This response will be treated by the system for choosing those entities that are related to the music field. A request for extra information will be done to the MusicBrainz database, accessing it through its Web service. The system will work to present this information in a friendly and attractive way to end users.

Keywords:

- Web Services
- Semantic Web
- OpenCalais
- MusicBrainz

Contenido

Índice de ilustraciones

Índice de tablas

1. Introducción y objetivos

1.1. *Introducción*

Los servicios Web surgen como respuesta a la necesidad de establecer un nuevo tipo de aplicaciones que sean más sencillas de integrar y sobre todo que esta integración sea independiente de la tecnología sobre la que estén construidas.

A partir de aquí surge la idea de servicio Web que se engloba dentro del término de la Web 2.0, que es una nueva Web con nuevas capacidades de interoperabilidad entre otras ventajas. Dada la importancia y el consecuente crecimiento que hoy en día tiene el mundo de los servicios Web y todo lo que les rodea, se plantea este proyecto como una oportunidad de indagar más en ellos.

Posteriormente surge el concepto de Web 3.0, que engloba, entre otros elementos, al concepto de Web Semántica. Este proyecto pretende acercar en un ámbito real la Web 2.0 y la 3.0 utilizando elementos propios de ambas, como lo son los servicios Web y la Web semántica.

El problema que se pretende resolver es por tanto la capacidad de un sistema para integrar diferentes servicios Web. Dentro del proyecto, esta idea se va a combinar con el concepto de Web semántica para dar lugar a un sistema que aprovecha las ventajas de los servicios Web para dotar de semántica a la información que se le indique.

El uso de OpenCalais como servicio Web aporta semántica a la información con la que se esté trabajando. Del amplio rango de contextos sobre los que trabaja este servicio Web, tales como información económica, cultural, política,… se ha considerado interesante enfocar su uso al contexto musical por tratarse de un contexto en el que este servicio Web está menos desarrollado y sobre todo porque es un ámbito de interés popular.

1.2. *Objetivos*

El objetivo principal de este proyecto es el de realizar una aplicación Web que sea capaz de llevar a cabo un análisis semántico de un texto concreto y además de dar la opción al usuario de ampliar la información de las entidades que se hayan reconocido durante este análisis.

Este objetivo se traduce en los siguientes subobjetivos:

1. El sistema debe ser capaz de realizar un reconocimiento semántico con la ayuda de un servicio Web especializado.

Esto permitirá además explorar el concepto de Web Semántica. Se ha planteado el uso de un servicio Web que hace las veces de analizador semántico, OpenCalais. Antes de realizar cualquier acción sobre el mismo se va a realizar un estudio de los principales conceptos que lo envuelven para entrar en el contexto de la Web Semántica y ser capaz de reconocer los términos que la describen.

Dada la amplitud de contextos sobre los que trabaja este servicio Web se ha considerado necesario limitar su uso para poder desarrollar una funcionalidad más específica. En concreto se ha establecido que se va a limitar su uso a contextos musicales, ya que es un tema de interés general que actualmente cuenta con muchas herramientas en la Web para su difusión y análisis.

2. Sistema capaz de ampliar la información de determinadas entidades, seleccionadas por el usuario de entre todas aquellas que hayan sido reconocidas durante la fase del análisis semántico. Esta información se va a conseguir a través de otro servicio Web externo.

   Dado que se ha decido acotar la capacidad de semántica de OpenCalais al ámbito de la música la elección de este segundo servicio Web será hecha dentro de los distintos servicios Web que aporten este tipo de información. En concreto se ha considerado oportuno realizar la segunda integración con el servicio Web de MusicBrainz, ya que se trata de un servicio Web que da acceso a información musical de una base de datos que es mantenida por los propios usuarios, lo que aporta la ventaja de que va a estar actualizada. Esto en el ámbito de la música es importante ya que surge nueva información constantemente.

3. Ya que va a ser requerida la ayuda de dos servicios Web completamente diferentes para aportar la funcionalidad descrita al sistema, va a ser necesaria la integración de estos servicios Web con el sistema.

   Para poder utilizar e integrar los servicios Web es necesario en primer lugar conocer sus conceptos principales, las arquitecturas sobre las que se pueden construir y sobre todo qué métodos de integración están disponibles para acceder a la funcionalidad que nos aportan. Por tanto en primer lugar se realizará un breve estudio sobre todos estos ámbitos relacionados con los servicios Web. Esto es sumamente necesario para poder realizar un uso adecuado de los servicios Web y de todo lo que nos ofrecen.

4. Al tratarse de dos servicios Web diferentes, será necesario además de integrarlos con el sistema, integrarlos entre sí para que la respuesta de uno de ellos se pueda traducir en una petición al otro.

   El sistema recogerá la información semántica procedente del servicio Web de OpenCalais realizando las transformaciones oportunas para realizar la petición de al servicio Web de MusicBrainz y aportar información que éste devuelva al usuario final del sistema de manera sencilla y con un componente visual que lo haga más atractivo.

1.3. *Fases del desarrollo*

Las fases del desarrollo de este proyecto han sido las siguientes:

1. Planificación: en primer lugar se realizó una estimación de los recursos humanos que se iban a necesitar, durante cuánto tiempo y con qué intensidad teniendo en cuenta las diferentes fases del desarrollo de un software.

2. Toma de requisitos: después se realizó un estudio de las principales características que debía tener el sistema y de la funcionalidad que iba a aportar el mismo.

3. Estudio del entorno: una vez establecido el comportamiento deseado del sistema, fue necesario realizar un estudio del entorno, de las tecnologías que iban a ser necesarias para el desarrollo del sistema. Es en este momento donde se indaga sobre los conceptos fundamentales sobre servicios Web y la Web semántica, así como de las tecnologías necesarias para implementar estos conceptos.

4. Análisis del sistema: una vez adquiridos estos conceptos, se analizó la solución del problema a resolver y las diferentes tecnologías que se podían emplear para la consecución de la misma, seleccionando aquellas que se complementan mejor y que describían la solución de manera más sencilla y eficiente.

5. Diseño del sistema: en este punto, ya se conocen las tecnologías a emplear y se ha desarrollado una arquitectura que se ajuste lo más posible a la de la solución ideal. Es entonces cuando se diseña la secuencia de operaciones a realizar para proporcionar la funcionalidad que se estableció durante los requisitos con la arquitectura escogida durante el análisis. Se divide el sistema en componentes agrupados según la funcionalidad que aportan.

6. Implementación de la solución: una vez se han establecido los componentes, se van desarrollando cada uno de ellos con las tecnologías y la arquitectura seleccionadas durante las fases anteriores.

7. Fase de pruebas: antes de la realización de las pruebas, es necesario el establecimiento de las mismas, estudiando las posibles debilidades del sistema para acabar con ellas en el caso de que sean debidas a fallos en el desarrollo del sistema.

1.4. *Medios empleados*

Los medios humanos con los que se ha contado para la realización de este proyecto es, en primer lugar, y bajo la supervisión de un jefe de proyecto, un equipo de desarrollo.

Respecto a los recursos materiales, se ha contado con un único equipo de entorno Windows que utiliza software de libre distribución para todas las labores de desarrollo del sistema, a excepción de un paquete Microsoft Office. El resto de software necesario se documentará en el apartado dispuesto para la descripción del entorno, tanto de desarrollo: **4.1Entorno de desarrollo**, como de ejecución: **4.10Entorno de ejecución.**

Se detallan estos recursos humanos y materiales en el **Anexo III: Presupuesto** en el que se incluye el presupuesto para este proyecto. También se acompaña de un **Anexo II: Planificación,** planificación del proyecto que estima las horas necesarias para la consecución del mismo y que ayuda a realizar el presupuesto del mismo.

1.5. *Estructura de la memoria*

A continuación se va a describir el contenido de este documento, aportando un breve resumen de cada uno de sus capítulos.

El capítulo 2, Web Semántica, está dedicado a poner en contexto al lector en el ámbito de la Web Semántica tratando de definir los principales concepto que están involucrados en la misma, haciendo énfasis en los que están más relacionados con el funcionamiento del sistema que ocupa a este proyecto.

El capítulo 3, Servicios Web, comienza con una descripción de lo que son los servicios Web y algunas definiciones que son necesarias para comprender el funcionamiento de los mismos, así como su integración con otros sistemas. Se introducirá al lector en el ámbito de los dos servicios Web externos de los que va a hacer uso este sistema y en las diferentes tecnologías en las que se ha apoyado durante su construcción.

En el capítulo 4, Descripción del Sistema, se explica el comportamiento del sistema desde su Arquitectura hasta las secuencias de uso. Se muestran los elementos que han ayudado a realizar el diseño del sistema para justificar el desarrollo del mismo. También se muestran las pruebas que ha pasado el sistema para asegurar su correcto funcionamiento.

El capítulo 5, Conclusiones, se analizan las conclusiones a las que se ha llegado tras la construcción del sistema y haber reflexionado sobre las capacidades del mismo. Además se analiza cuál es la continuidad y evolución del sistema a lo largo del tiempo estudiando las futuras líneas de trabajo.

En el capítulo 6, Referencias bibliográficas y material consultado, se indican qué material bibliográfico y otras fuentes de datos se han consultado para llevar a cabo la construcción de este sistema.

En los anexos se incluye un manual de puesta en marcha para ayudar en la instalación del sistema y un manual de usuario con las instrucciones de uso del mismo.

1.6. *Acrónimos*

A continuación se listan alfabéticamente una serie de acrónimos que aparecerán a lo largo de este texto, muchos de ellos repetidos:

- **Ajax:** Asynchronous JavaScript And XML (JavaScript asíncrono y XML).

- **DOM:** Document Object Model (Modelo en Objetos para la representación de Documentos).
- **DTD:** Document Type Definition (Definición de tipo de documento).
- **JSON:** JavaScript Object Notation (Notación de Objetos JavaScript).
- **NPL:** Natural Language Processing (Procesamiento del lenguaje natural).
- **PHP:** acrónimo recursivo que significa PHP Hypertext Pre-processor (Preprocesador de hipertexto PHP)
- **REST:** Representational State Transfer (Transferencia de Estado Representacional)
- **RDF:** Resource Description Framework (Marco de descripción de recursos)
- **RPC:** Remote Procedure Calls (Llamada a procedimiento remoto)
- **SOAP:** Simple Object Access Protocol (Protocolo simple de acceso a objetos)
- **UDDI:** Universal Description, Discovery and Integration (Descripción, descubrimiento e integración universal)
- **URI:** Uniform Resource Identifiers (Identificador uniforme de recurso)
- **URL:** Uniform Resource Locator (Localizador uniforme de recursos)
- **W3C:** World Wide Web Consortium (Consorcio World Wide Web)
- **WSDL:** Web Service Description Language (Lenguaje de descripción de servicios Web)
- **XLL:** eXtensible Linking Language (Lenguaje extensible de enlaces)
- **XML:** Extensible Markup Language (Lenguaje extensible de marcas)
- **XSL:** eXtensible Stylesheet Language (Lenguaje extesible de hojas de estilo)
- **XUL:** XML User Interface Language (Lenguaje XML de interfaz de usuario)

2. Web semántica

Este apartado está dedicado a la introducción de los conceptos principales sobre los que se ha basado este proyecto.

Básicamente los dos pilares conceptuales de este proyecto son la Web Semántica y los Servicios Web, por lo que ambos serán definidos, así como otras tecnologías que puedan influir en los mismos.

A continuación se tratan los conceptos básicos de la Web Semántica, ya que este proyecto utiliza un Servicio Web que está basado en la misma y es necesario entrar un ligeramente en el contexto.

2.1. *¿Qué es la Web semántica?*

La Web semántica es una extensión de la Web tal y como la conocemos, que se encarga de dotar a la misma de un mayor significado. Está desarrollada con lenguajes universales que permiten a los usuarios encontrar respuestas a sus preguntas de una forma más rápida y sencilla gracias a la mejor estructuración de la información. Esto se consigue gracias a que los recursos están anotados de forma que los ordenadores pueden comprender la función o servicio que ellos mismos prestan.

La problemática bajo la cual nace la Web Semántica se basa en dos puntos importantes:

- La inexistencia de mecanismos que procesen de manera automática toda la información que se va almacenando a lo largo del tiempo en la gran cantidad de páginas Web.
- Los problemas derivados de la falta de interoperabilidad entre aplicaciones.

Si se indaga un poco en estos problemas en seguida sale a flote la necesidad de la Web Semántica.

En la Web actual tenemos un sistema que almacena, prácticamente, toda la información del mundo y que nos permite acceder a él de manera casi automática desde cualquier lugar del planeta, simplemente con una conexión a Internet. La información que consultamos puede estar a su vez conectada con otra a la que referencia desde ella misma, es decir, puede redirigirnos a otros sitios Web, a otro tipo de documentos, imágenes, etc. Para ayudarnos con esto han aparecido numerosos buscadores, como Google, que indexa millones de páginas, pero a pasar de ese esfuerzo están lejos de mostrar al usuario lo que realmente busca. Esto es debido a que no son capaces de enlazar con la totalidad de páginas existentes, que los resultados encontrados pueden no ser muy precisos e incluso que para realizar una búsqueda cien por cien eficaz haya que introducir las palabras exactas.

A este problema derivado de la falta de semántica a la hora de realizar las búsquedas, se añade el de la falta de interoperabilidad, que se resuelve también con la Web Semántica al establecerse determinados marcos comunes que facilitan la comunicación entre aplicaciones.

Como dice la propia página de la W3C sobre la Web Semántica, que proporciona un framework común que permite que se puedan compartir y reutilizar los datos a través de los límites impuestos únicamente por las empresas, aplicaciones o comunidad. Es un esfuerzo de desarrollo liderado por la W3C y acompañado del de un gran número de investigadores y socios industriales.

Es decir se trata de que la propia Web en sí sea capaz de ser consciente de la información que alberga y del significado de la misma, tal y como hace un ser humano.

La Web semántica se basa en la idea de añadir metadatos semánticos y datos ontológicos a la Web. Toda esta información adicional describe el contenido, el significado y la relación entre datos y para que pueda ser evaluada automáticamente por máquinas de procesamiento se debe proporcionar de manera formal.

El objetivo con el que nace la Web semántica, o también llamada Web 3.0, es mejorar Internet ampliando la interoperabilidad entre los sistemas informáticos y reducir mediación de operadores humanos.

2.2. Ontologías

La idea de ontología no es nueva, es un término que ya era utilizado por la civilización griega. Aristóteles lo definía como la ciencia del ser. Actualmente en el ámbito de la Web Semántica, simplemente se define como conjunto de vocablos que describen un dominio determinado.

El concepto más importante para hacer posible la Web Semántica es tal vez la Ontología. Es necesario que todo este conocimiento sea legible por los ordenadores y se represente por un lenguaje que esté unificado y además se pueda reutilizar. Las ontologías proporcionan esta unicidad para representar el conocimiento ya que proveen de comprensión del conocimiento de un dominio de manera compartida y consensuada para que así pueda ser transmitida entre personas y sistemas heterogéneos.

Las ontologías representan este conocimiento a través de los siguientes componentes:

- Los conceptos: o ideas que se intentan formalizar.
- Las relaciones: que representan la interacción entre los conceptos.
- Las funciones: relación concreta que identifica un elemento mediante el cálculo de una función que considera varios elementos de la ontología.
- Las instancias: representación de objetos determinados.
- Axiomas: teoremas que se declaran sobre las relaciones que deben cumplir los elementos de la ontología.

De esta manera se organiza la información para que los sistemas puedan interpretar su significado y por tanto buscar e integrar datos de manera más eficiente. Las aplicaciones

podrán ir más allá del simple volcado de datos e interpretar los datos de las páginas Web sacando conclusiones de los mismos. Así una simple búsqueda queda mejorada al aportar la semántica necesaria para que los resultados mostrados realmente se correspondan con lo buscado y no se traten de una simple aproximación sintáctica.

Con la aparición de este concepto surge la necesidad de crear diversos medios o lenguajes para la correcta descripción de las ontologías. Para ello nace OWL, bajo la necesidad de que este tipo de información necesita ser procesada por las aplicaciones y no ser meramente representada. OWL se basa en la definición de etiquetas de XML y en la representación de los datos de RDF.

2.3. *Capas de la Web Semántica*

W3C representa con en este gráfico, los principios sobre los que está implementada la Web Semántica:

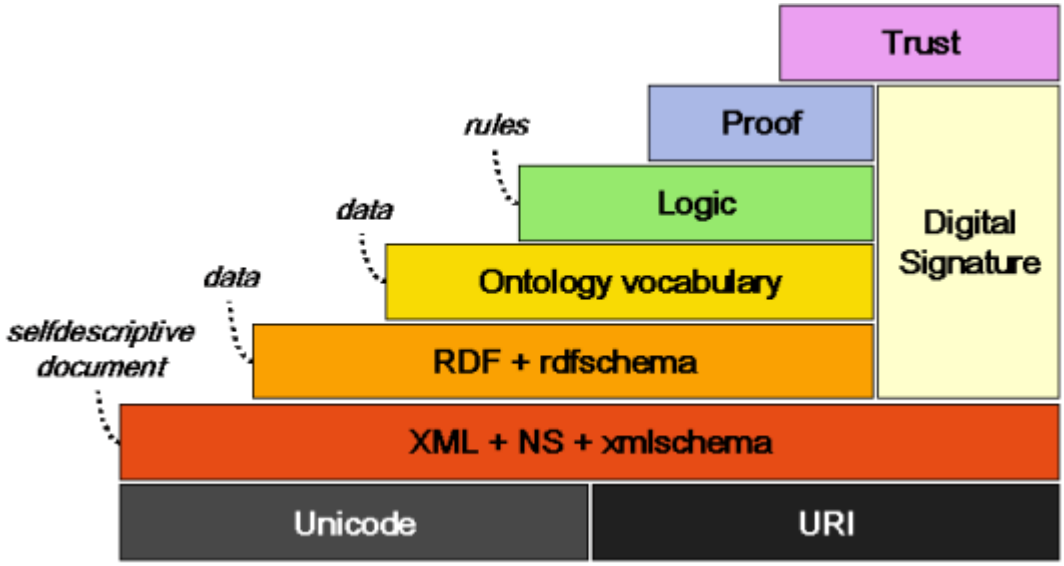


Ilustración 1: Capas de la Web Semántica [6]

- **Unicode:** Este estándar permite que cualquier texto de cualquier idioma, pueda ser codificado y utilizado con usos informáticos, a pesar de que aparezcan en él caracteres o símbolos extraños.

- **URI:** Son cadenas que permiten acceder a cualquier recurso de la Web. En la Web Semántica las URIs son las encargadas de identificar objetos.

- **XML+NS+xmlschema:** Esta es la capa más técnica de la Web Semántica. En ella se encuentran agrupadas las diferentes tecnologías que posibilitan la comunicación entre

agentes asegurando que se pueda integrar la Web Semántica con otros estándares basados en XML.

El XML nos ofrece un formato común para el intercambio de documentos, Namespaces (NS) proporciona un método para cualificar elementos y atributos de nombres usados en documentos XML asociándolos con espacios de nombre identificados por referencias URIs. XML Schema es un lenguaje que permite describir la estructura y restringir el contenido de documentos XML.

- **RDF+rdfschema:** RDF surge como respuesta a la necesidad de intercambiar información entre aplicaciones sin que ésta información pierda su sentido durante el intercambio. Nos proporciona un framework para poder expresar esta información.

  Se basa en la idea de representar los objetos en la Web a través de identificadores únicos (URIs, Uniform Resource Identifiers) y de describir los recursos en términos de expresiones formados por tripletes sujeto-predicado-objeto, en los que el sujeto es el recurso, lo que se está describiendo; el predicado, la propiedad o relación que se desea establecer acerca del recurso y el objeto, el valor de la propiedad. O lo que es lo mismo, un triplete recurso-propiedad-valor.

  Esta capa está basada en la capa anterior, define el lenguaje universal con el que podemos expresar diferentes ideas en la Web Semántica. RDF es un lenguaje que define un modelo de datos para describir recursos mediante tripletas sujeto-predicado-objeto.

  Los dos primeros serán URIs y el tercero puede ser URI o un valor literal. RDF Schema es un vocabulario RDF que nos permite describir recursos mediante una orientación a objetos. Esta capa no sólo ofrece una descripción de los datos, sino también cierta información semántica.

- **Ontology (Ontologías):** Nos permite clasificar la información. Esta capa permite extender la funcionalidad de la Web Semántica agregando nuevas clases y propiedades para describir los recursos.

- **Logic (Lógica):** Además de ontologías se precisan reglas de inferencia.

- **Proof (Pruebas):** Se intercambiarán “pruebas” escritas en el lenguaje unificador de la Web Semántica. Este lenguaje posibilita las inferencias lógicas realizadas a través del uso de reglas de inferencia.

- **Trust (Confianza):** Hasta que no se haya comprobado de forma exhaustiva las fuentes de información, los agentes deberían ser muy escépticos acerca de lo que leen en la Web Semántica.

- **Digital Signature (Firma digital):** Es utilizada por los ordenadores y agentes para verificar que la información ha sido ofrecida por una fuente de confianza y que no ha sido modificada por ningún agente externo. Es decir, asegura la integridad de la información.

3. Servicios Web

El siguiente de los conceptos principales tratados en este proyecto es el de Servicio Web. A continuación se realiza una descripción de qué son, cómo funcionan y cuáles son las formas de acceso a lo que nos ofrecen. También se realiza un resumen de las capacidades de los servicios Web en los que se apoya este proyecto y cómo se utilizan.

3.1. *¿Qué son?*

Si se coge el término servicios Web y se toma al pie de la letra, su significado sería que tratan de aplicaciones que prestan servicio o que ofrecen su funcionalidad a través de la Web. Esta definición es demasiado abierta y no diferencia entre un auténtico servicio Web, por ejemplo, una página Web cualquiera.

Se puede conseguir una definición más técnica consultando a W3C, que dice que se trata de una aplicación software identificada por una URI, cuyas interfaces y conexiones son capaces de ser definidas, descritas y descubiertas como artefactos XML. Es una manera estandarizada de integrar aplicaciones basadas en la Web utilizando los estándares XML, SOAP, WSDL y UDDI.

Resumiendo un poco para qué se utilizan cada uno de estos estándares, XML se utiliza para etiquetar los datos, SOAP para transportarlos, WSDL como descriptor de los servicios que presta un servicio Web concreto y UDDI para listar los servicios Web disponibles.

Un servicio Web soporta interacción directa con otros módulos de software mediante el intercambio de mensajes basados en XML que se envían a través de protocolos basados en Internet. Uno de los propósitos de los servicios Web es la reutilización a nivel global de diversas soluciones propuestas por otros. Por lo tanto podríamos considerar que los requisitos o características que tiene que cumplir un servicio Web para poder considerarse como tal son:

- Lo primero es que debe estar basado en Internet y prestar servicio a través del mismo. Por lo tanto debe utilizar sus estándares y protocolos.
- La interoperabilidad, ya que debe prestar su uso a clientes externos de diversas plataformas.
- Utilizar los estándares XML, SOAP, WSDL y UDDI. Lo que permitirá a sus clientes la utilización de prácticamente cualquier lenguaje para acceder externamente a los mismos.

En estas definiciones y requisitos se puede observar que hay unos denominadores comunes que aparecen repetidamente debido a su importancia y a que son lo que realmente constituye a los servicios Web.

A continuación se muestra la arquitectura extraída del Sitio de W3C, que representa un esquema muy completo, en el que se puede apreciar que dependiendo del tipo de cliente que esté demandando la información se utilizarán unos estándares u otros, pero las capas básicas que forman el servicio Web son comunes a todos, consiguiendo, como se decía que era fundamental, interoperabilidad entre sistemas o aplicaciones completamente heterogéneas.

Ilustración 2: Capas de un Servicio Web [6]

Por lo tanto la manera que cualquier aplicación tendría de comunicar con un servicio Web sería la siguiente:

1. Se descargaría el fichero de descripción del servicio (WDSL), para recabar información acerca de cómo interactuar con él. Este fichero de descripción del servicio Web implica una serie de meta datos estructurados sobre la interfaz que intenta utilizar la aplicación cliente así como documentación escrita sobré el servicio Web pudiendo incluir algún ejemplo de uso.

2. Para poder intercambiar datos, cliente y servidor tienen que disponer de un mecanismo común de codificación y un mismo formato de mensaje. Este sería el cometido del protocolo SOAP de intercambio de mensajes.

3. Una vez el cliente conoce la estructura del mensaje puede escribirlo. Los datos encapsulados dentro de este mensaje tienen que seguir el esquema de XML. Esto asegura la unicidad a la hora de recibir una respuesta ya que si cada servicio Web codificara la información a su manera no se podría acceder de manera estandarizada a la información que nos prestan.

4. Cuando se dispone del mensaje formateado según el protocolo debido, se transfiere entre cliente y servidor mediante algún protocolo de transporte.

A continuación se muestran más en detalle las tecnologías utilizadas en cada una de estas capas, lo que nos va a ayudar a definir una manera más técnica y detallada a los servicios Web.

WSDL:

Como se ha mencionado con anterioridad, cualquier cliente que demande un servicio de un servicio Web necesita primero informarse de cómo está estructurada la información que proporciona. Esta es la función esencial del WSDL, describir la información que se va a servir, cómo son los mensajes que va a precisar durante la comunicación, mediante qué protocolo se precisa su posterior envío, etc. es decir establecer un marco común de comunicación con el cliente. Todo esto basándose en XML para la descripción.

Dado que los protocolos de comunicaciones y los formatos de mensajes están estandarizados, cada día aumenta la posibilidad e importancia de describir las comunicaciones de forma estructurada. WSDL afronta esta necesidad definiendo una gramática XML que describe los servicios de red como colecciones de puntos finales de comunicación capaces de intercambiar mensajes. Las definiciones de servicio de WSDL proporcionan documentación para sistemas distribuidos y sirven como fórmula para automatizar los detalles que toman parte en la comunicación entre aplicaciones.

Un documento WSDL utiliza los siguientes elementos en la definición de servicios de red:

- **Types**: contenedor de definiciones del tipo de datos que utiliza algún sistema de tipos (por ejemplo XSD). Contiene información de esquema referenciado en el documento WSDL. El sistema de tipos predeterminado que admite WSDL es de esquema de XML. Si se usa esquema de XML para definir los tipos que contiene el elemento Types el elemento Schema aparecerá inmediatamente como elemento hijo.

- **Message**: definición abstracta y escrita de los datos que se están comunicando. Proporciona una abstracción común para el paso de mensajes entre el cliente y el servidor. Pueden aparecer, múltiples elementos Message en un documento WSDL, uno para cada mensaje que se comunica entre el cliente y el servidor.

- **Operation**: descripción abstracta de la acción admitida por el servicio.

- **Port Type**: conjunto abstracto de operaciones admitidas por uno o más puntos finales, es decir, representan los tipos de correspondencia que pueden producirse entre el cliente y el servidor, que puede ser: Request-response (petición-respuesta), One-way (un-sentido), Solicit-response (solicitud-respuesta) o Notification (Notificación).

- **Binding**: especificación del protocolo y del formato de datos para un tipo de puerto determinado. También indican el número de comunicaciones de re red que se requieren para realizar una determinada acción.

- **Port**: punto final único que se define como la combinación de un enlace y una dirección de red.

- **Service**: colección de puntos finales relacionados, es decir, grupo de puertos relacionados y a los que se hace referencia por una dirección única.

SOAP

Se trata de un protocolo para empaquetar llamadas remotas a procedimientos, del inglés, Remote Procedure Calls (RPC), muy simple, ya que el usuario no tiene por qué comprender todo el contenido de una llamada, puede ignorar partes del XML y quedarse sólo con las que le interesen.

Se codifica en XML, utilizando XML Schema y XML Namespaces. En el siguiente apartado se trata más en detalle la construcción o empaquetado de este tipo de mensajes.

XML

XML es el lenguaje de marcas sobre el que se asientan los servicios web. Los lenguajes de marcas no son equivalentes a los lenguajes de programación, aunque se definan también como lenguajes, son sistemas de descripción de información.

Existen tres utilidades básicas de los lenguajes de marcas: describir su contenido, definir su formato o realizar las dos funciones indistintamente. Las aplicaciones de bases de datos son buenas referencias del primer sistema, los programas de tratamiento de textos son ejemplos típicos del segundo tipo, y el HTML es la muestra más conocida del tercer modelo.

XML no es más que un conjunto de reglas para definir etiquetas semánticas que nos organizan un documento en diferentes partes, un metalenguaje que define la sintaxis utilizada para definir otros lenguajes de etiquetas estructurados.

El metalenguaje XML consta de cuatro especificaciones

- **DTD:** Archivo que encierra la definición formal y, a la vez, especifica la estructura lógica de cada documento. Define tanto los elementos de una página como sus atributos.

- **XSL:** Define o implementa el lenguaje de estilo de los documentos escritos para XML, esto permite modificar el aspecto de un documento.

- **XLL:** Define el modo de enlace entre diferentes enlaces. Va más allá de los enlaces simples que sólo soporta el HTML. Se podrá implementar con enlaces extendidos.

Como se indica a continuación, su uso encaja a la perfección con la idea de servicio Web, ya que ayuda a estos últimos a seguir las características que acompañan al concepto en sí de servicio Web:

- Se trata de un estándar abierto, es decir que es reconocido mundialmente, por lo que la gran mayoría de software permite la compatibilidad con XML. Esto lo que lo hace muy potente a la hora de establecer comunicación entre distintas plataformas de software y hardware, que como se ha comentado con anterioridad, es el sentido final de los Servicios Web.

- También es una ventaja la simplicidad de sintaxis. Es muy fácil escribir código en XML y su representación de los datos es casi entendible por cualquier ser humano. Esto lo hace muy flexible a la hora de querer representar datos de cualquier tipo y ámbito y lo convierte en el lenguaje ideal para utilizar en los servicios Web.

- Es independiente del protocolo de Transporte, solo necesita de uno que pueda transferir texto o documentos (HTTP, SMTP,..). Como se ha comentado al principio de esta apartado, una de las características de los servicios Web, es la independencia del protocolo de trasporte, con lo que también encaja con ésta.

3.2. *Tipos de Servicios Web*

Básicamente hay dos tipos de servicios Web si los clasificamos por el acceso que se puede hacer a los mismos, los Servicios Web basados en SOAP y los Servicios Web basados en REST. A continuación se detallan ambos tipos y las diferencias que existen entre ellos.

3.2.1. Servicios Web SOAP

A la hora de realizar la comunicación con el Web Service, una opción que se puede extraer del apartado de definición es la utilización del protocolo SOAP, es decir, realizar un intecambio de mensajes empaquetados siguiendo el protocolo SOAP.

A continuación se muestra cómo son estos mensajes con un ejemplo visual extráido del sitio de W3C:

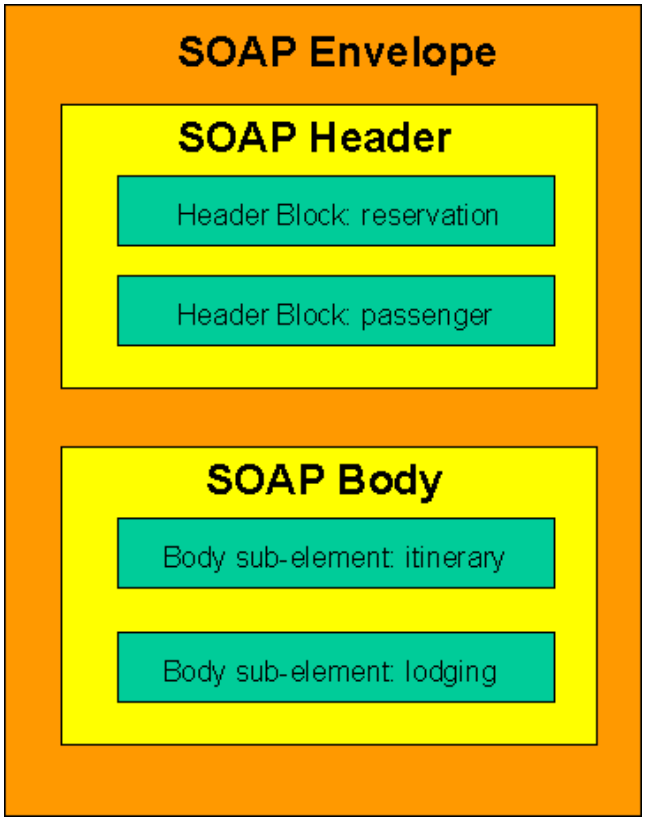


Ilustración 3: Estructura de un mensaje SOAP [6]

El elemento raíz del documento es el elemento Envelope, que es el sobre del mensaje, por lo que contendrá toda la información a comunicar. Este sobre puede contener un elemento Header que contiene información sobre el mensaje, en concreto puede estar compuesto por dos elementos, uno que describa a quién compuso el mensaje y otro el posible receptor del mismo.

Lo que sí debe contener el sobre, es un elemento body (cuerpo), que contendrá los datos del mensaje, es decir el contenido que se desea transmitir. Este mensaje debe estar dentro de un sobre de SOAP bien construido.

Es decir el cuerpo contiene la carga de datos del mensaje y la cabecera contiene los datos adicionales que no pertenecen necesariamente al cuerpo del mensaje.

Además de definir el sobre de SOAP, la especificación de SOAP define una forma de codificar los datos contenidos en un mensaje, proporcionando un patrón de mensaje estándar para facilitar el comportamiento de tipo RPC.

Este sería el contenido de un mensaje SOAP, en el que se pude observar la estructura comentada. De hecho este ejemplo, extraído también del sitio de W3C, es la representación del que se muestra en la imagen anterior:

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
 <env:Header>
  <m:reservation xmlns:m="http://travelcompany.example.org/reservation"
                 env:role="http://www.w3.org/2003/05/soap-envelope/role/next"
                  env:mustUnderstand="true">
   <m:reference>uuid:093a2da1-q345-739r-ba5d-pqff98fe8j7d</m:reference>
   <m:dateAndTime>2001-11-29T13:20:00.000-05:00</m:dateAndTime>
  </m:reservation>
  <n:passenger xmlns:n="http://mycompany.example.com/employees"
                 env:role="http://www.w3.org/2003/05/soap-envelope/role/next"
                  env:mustUnderstand="true">
   <n:name>Åke Jógvan Øyvind</n:name>
  </n:passenger>
 </env:Header>
 <env:Body>
  <p:itinerary
    xmlns:p="http://travelcompany.example.org/reservation/travel">
   <p:departure>
    <p:departing>New York</p:departing>
    <p:arriving>Los Angeles</p:arriving>
    <p:departureDate>2001-12-14</p:departureDate>
    <p:departureTime>late afternoon</p:departureTime>
    <p:seatPreference>aisle</p:seatPreference>
   </p:departure>
   <p:return>
    <p:departing>Los Angeles</p:departing>
    <p:arriving>New York</p:arriving>
    <p:departureDate>2001-12-20</p:departureDate>
    <p:departureTime>mid-morning</p:departureTime>
    <p:seatPreference/>
   </p:return>
  </p:itinerary>
  <q:lodging
   xmlns:q="http://travelcompany.example.org/reservation/hotels">
   <q:preference>none</q:preference>
  </q:lodging>
 </env:Body>
</env:Envelope>
```

Ilustración 4: Mensaje SOAP de ejemplo [6]

Una de las opciones barajadas para realizar la comunicación con los servicios Web que se desean utilizar en este sistema, de los que se hablará a continuación, sería esta. Bastaría con desarrollar un componente que construyera mensajes SOAP con la estructura y datos necesarios para obtener la información deseada. El mismo componente tendría la labor de extraer esta información de mensaje de respuesta enviado de vuelta por el servicio Web en concreto.

3.2.2. Servicios Web REST

REST en realidad no es un protocolo, si no un estilo de arquitectura. Se trata de un término introducido por Roy Fielding en el año 2000 como resultado de su tesis doctoral, para describir un estilo de arquitectura software para los sistemas distribuidos de la Web.

Esta arquitectura sigue una serie de principios:

- El protocolo cliente/servidor no tiene estado, cada mensaje HTTP contiene toda la información necesaria.

- Define un conjunto de operaciones, las más importantes son POST, GET, PUT y DELETE.
  - GET se usa para conseguir datos o realizar consultas sobre un recurso. El servicio Web devuelve una representación del recurso requerido.

  - POST se usa para crear nuevos recursos. El servicio Web puede responder tanto con datos como con el estado éxito o fracaso.

    - PUT se usa para actualizar recursos o datos existentes.
    - DELETE se usa para borrar recursos o datos

- Dispone de una sintaxis universal para identificar los recursos, esto es a través de las URI.

- Se pueden intercalar componentes intermedios entre el cliente y los recursos como servidores Proxy, gateways, etc.

Actualmente se ha ampliado la concepción de REST a la de cualquier interfaz Web que utilice XML y HTTP sin necesidad utilizar un protocolo de servicios Web como SOAP.

El funcionamiento de esta arquitectura es muy sencillo, el cliente hace una petición al servidor a través de HTTP indicando una de las operaciones citadas en los principios de REST y el servidor le contesta.

A continuación se explican los dos métodos que se utilizan en los servicios Web REST para realizar las peticiones de información al servidor:

GET

Método de petición de información mediante HTTP que se utiliza para recuperar información que es identificada por un URI. El método GET también se puede utilizar para pasar una pequeña cantidad de información al servidor en forma de pares atributo-valor añadidos al final del URI detrás de un símbolo de interrogación. Por ejemplo en llamadas a servicios Web que requieran el uso de parámetros de entrada estos parámetros se le indicarán al servidor a través de la URI en pares atributo=valor, separándolos de la URL del servicio Web mediante una interrogación y, en el caso de que haya varios, entre ellos mediante un el ampersand.

POST

Este método normalmente invocará procesos que generan datos que se van a devolver como respuesta a la petición, además puede aportar datos de entrada a estos programas. Los pares atributo-valor son incluidos en el cuerpo de la petición separados por *ampersand* y por lo tanto no viajarán de manera visible en la URL que se le indique al navegador. Además de no resultar visibles a primera vista permiten el envío de datos más grandes que lo que se pueden enviar a través del método GET.

3.2.3. Diferencias

A través de algunas de sus características se pueden apreciar algunas de las principales diferencias entre los servicios Web basados en SOAP y los basados en REST:

Características de REST:

- ✓ Las operaciones se definen en los mensajes que se envían al servidor.
- ✓ Una dirección única para cada instancia del proceso.
- ✓ Cada objeto soporta las operaciones estándares definidas.
- ✓ Componentes débilmente acoplados.
- ✓ Transporte http
- ✓ El servidor no tiene estado.
- ✓ Se pueden añadir sesiones a través de las cookies.
- ✓ Seguridad en la comunicación a través de HTTP seguro (HTTPS).
- ✓ Necesita de un documento descriptivo en el que se indique cómo acceder a los recursos y cómo va a responder el servicio Web.

Características de SOAP:

- ✓ Las operaciones se definen como puertos WSDL.
- ✓ Hay una única dirección para todas las operaciones.
- ✓ Múltiple instancias del proceso comparten la misma operación.
- ✓ Componentes fuertemente acoplados.
- ✓ Independiente del transporte.
- ✓ El servidor puede almacenar el estado de la conversación.
- ✓ Las sesiones se añaden en el propio mensaje en la llamada cabecera de sesión.
- ✓ Seguridad en la comunicación a través del protocolo de comunicaciones seguras en los servicios Web (WS-Security).
- ✓ Para la descripción de su uso tiene que poner a disposición el WSDL.

3.3. *Servicio Web de OpenCalais*

El primero de los Servicios Web que se han utilizado para este proyecto es el desarrollado a través de la iniciativa OpenCalais. A continuación se explica qué hace este servicio y de qué manera se puede conseguir la información, además de cómo la va a ofrecer este servicio.

3.3.1. ¿Qué es?

OpenCalais es una iniciativa, que nace en enero de 2008 de la mano de Thomson Reuters para fomentar el desarrollo de las tecnologías semánticas en la Web.

El *OpenCalais Web Service* es un servicio Web que permite enriquecer con metadatos semánticos el contenido que se le indique. En concreto es capaz de distinguir a través de un análisis semántico determinadas entidades como personalidades, compañías, eventos tales como adquisiciones o cambios organizativos, etc. Fue creado especialmente para este tipo de contextos, es decir está más orientado al mundo de los periódicos y las noticias

El servicio Web de OpenCalais recibe información desestructurada y devuelve resultados formateados identificando las entidades, los hechos y los eventos que aparecen en el texto.

El procesamiento que hace es completamente automático, Calais aprende usando reglas. Lo que hacen para ello es intentar reproducir el modo en que un humano al leer identifica y rompe la ambigüedad de los términos. Tal y como hace el humano, mientras está leyendo un texto, usa las pistas que el contexto le aporta, utilizando para ello el contexto cercano y el contexto global.

Han desarrollado un sistema basado en reglas utilizando para ello un lenguaje de programación propietario. Estas reglas están basadas en varios niveles NLP, desde la agrupación del texto en unidades al análisis morfológico y léxico para distinguir entre nombres y verbos.

Actualmente se están dedicando a ampliar estas reglas de manera que las abreviaturas, los acrónimos y los nombres incompletos se puedan reconocer igualmente, pero de momento esto es una limitación.

3.3.2. ¿Cómo se consigue la información?

Para poder acceder al servicio de reconocimiento semántico que presta OpenCalais es necesario saber cómo se puede acceder a la información, es decir qué métodos pone OpenCalais a nuestra disposición para tener acceso a la funcionalidad que presta. Actualmente hay varios métodos para comunicarse con este servicio Web y así optar a sus servicios, pero la lógica es la misma para todos ellos, aunque los medios sean distintos.

Básicamente, para poder realizar el reconocimiento semántico de un texto, OpenCalais necesita un número licencia (es gratuito pero hay que registrarse), el texto que se desea analizar y una serie de parámetros de configuración para establecer elementos como el formato de la respuesta o el tipo de petición.

Nos encontramos por tanto con los siguientes métodos para la comunicación con este servicio Web:

El primero es a través de SOAP, protocolo del que ya se habló en este mismo capítulo, al definir los Servicios Web. OpenCalais proporciona un api que dispone de un método para realizar este tipo de llamadas. A este método hay que aportarle la información que comentábamos, la licencia para utilizar el servicio Web, que es una cadena alfanumérica, el contenido que se desea analizar y los parámetros en formato XML para indicar las opciones:

String Enlighten (String licenseID, String content, String paramsXML)

Como se vio en la definición del protocolo SOAP y de los servicios Web para realizar este tipo de llamadas es necesario consultar el WSDL con la descripción del servicio para conocer el formato de estos mensajes SOAP que se van a intercambiar. Se encuentra en: http://api.opencalais.com/enlighten/?wsdl. Y la URL a la que se le va a demandar el servicio es: http://api.opencalais.com/enlighten/.

Otro método de acceso es a través de REST, del que también se habló al principio del capítulo, para el cual están habilitadas las operaciones GET y POST para comunicarnos con este servicio

Web. La URL http://api.opencalais.com/enlighten/rest/ es a la que hacer este tipo de llamadas, indicándole la información debida que se comentaba, pero a través de una cadena compuesta por la lista de parámetros en pares de nombre y valor.

```
licenseID=url-encoded-string&content=url-encoded-string&paramsXML=url-encoded-string
```

Como se decía, la licencia, el contenido a analizar y los parámetros en XML.

También se puede comunicar a través de http Traffic Compression, para conseguir una respuesta Gzipped. Pero es necesario tener un conocimiento adecuado de este tipo de formato.

Los parámetros que se le tienen que indicar al servicio Web de OpenCalais le van a aportar información tan necesaria como el formato en que se está haciendo la llamada, es decir qué tipo de contenido le va a llegar y en qué formato se desea que se devuelva la respuesta. Además hay otra serie de parámetros configurables que van a especificar qué se desea que se devuelva, es decir si se desea información extra como la relevancia de los términos analizados dentro del contexto. Todos estos parámetros se pueden consultar en detalle en la Web:

http://www.opencalais.com/documentation/calais-web-service-api

3.3.3. ¿Cómo se devuelve la información?

Como se ha podido comprobar en el formato de la petición, existe un parámetro que indica en qué formato se desea que OpenCalais Web Service devuelva la respuesta.

En concreto estos son los posibles formatos de devolución:

- **RDF**: esta respuesta incluye información general del documento además de información transaccional, como el idioma del documento, fecha de la petición, ID de la petición. También incluye el contenido de entrada después de que se haya transformado en un documento XML válido tras el procesado de OpenCalais (excepto si se ha seleccionado la opción TEXT/RAW).

  La cabecera RDF contiene un resumen de todas las entidades extraídas del texto ordenadas alfabéticamente y organizadas según los tipos de Entidad.

- **Microformatos**: este formato de respuesta soporta los siguientes microformatos: rel-tag, hCard and hCalendar. Esta respuesta es devuelta como un stream de texto que se pude insertar directamente en el código HTML de cualquier página sin afectar a su visualización. Las entidades o eventos reconocidos se pueden visualizar subrayados por navegadores preparados para ello. (Se utiliza la etiqueta en la respuesta).

- **Formato simple**: proporciona una visión rápida del contenido RDF y facilita el procesado e integración de los datos devueltos.

- **JSON**: una alternativa a la respuesta en XML, ya que es más compacto y presenta simplicidad a los usuarios que estén acostumbrados a JavaScript.

3.4. *Servicio Web de MusicBrainz*

El otro servicio Web que se ha utilizado para este proyecto es el desarrollado por *MusicBrainz*. A continuación se explica qué hace este servicio y de qué manera se puede conseguir la información, además de cómo va a ser ofrecida por este servicio.

3.4.1. ¿Qué es?

MusicBrainz es una comunidad abierta que recolecta metadatos relacionados con la música en forma de base de datos relacional. Todos estos datos son mantenidos por los usuarios.

El modelo de datos que sigue esta base de datos es orientado a objetos y sus clases principales son Artistas, Discos, Canciones y Etiquetas, cada una d de las cuales tiene una serie de atributos y de relaciones entre ellos. Su base de datos está construida en PostgreSQL, que es un sistema gestor de base de datos relacional orientada a objetos y que es de código abierto.

Estos son los datos que almacena de cada tipo de objetos, según lo que se puede leer en el sitio de MusicBrainz que está dedicado a la descripción de su base de datos:

Artistas:

- Un identificador en la base de datos de MusicBrainz (MBID)
- Nombre
- Nombre corto para poder mostrar al artista cómodamente en las listas ordenadas.
- Alias que se utilicen comúnmente para este artista.
- El tipo, si es una persona o un grupo
- Fecha de inicio, que es la de nacimiento si es una persona o la de inicio del grupo
- Fecha de fin, fecha de la muerte o del fin del grupo dependiendo de si es una persona o un grupo.
- Comentarios, información del artista que puede ser útil para desambiguar
- Anotaciones, campo de texto de escritura libre.

Grupo de Discos: -concepto que aparece para representar de manera conjunta a un grupo de discos como si se trataran de una sola entidad-

- Un identificador en la base de datos de MusicBrainz (MBID)
- Título
- Artista

- Tipo, si se trata de un álbum, un single, una recopilación, un concierto, una entrevista, otros.

Discos:

- Un identificador en la base de datos de MusicBrainz (MBID)
- Título
- Artista
- Tipo, si se trata de un álbum, un single, una recopilación, un concierto, una entrevista, …
- Estado, si es oficial, está de promoción, etc.
- Idioma
- Anotaciones, campo de texto de escritura libre.
- Disc ID (zero or more disc IDs that allow audio CD identification)
- Identificador del disco en Amazon
- Eventos del lanzamiento:
    - Fecha del lanzamiento
    - País del lanzamiento
    - Etiqueta
    - Número de catálogo
    - Código de barras (EAN/UPC)
    - Formato (CD, cassette, vinyl, wax cylinder, etc.)

Canciones:

- Un identificador en la base de datos de MusicBrainz (MBID)
- Título
- Artista
- Duración (en milisegundos)
- Anotaciones, campo de texto de escritura libre.
- ISRC

Sellos o compañías musicales:

- Un identificador en la base de datos de MusicBrainz (MBID)
- Nombre
- Nombre corto descriptivo para mostrar con comodidad en las listas
- Alias
- Tipo (original production, bootleg production, reissue production, distributor, holding)
- Code, the IFPI label code
- Fecha de inicio
- Fecha de fin
- Comentarios, información del artista que puede ser útil para desambiguar
- Anotaciones, campo de texto de escritura libre.
- Países

El servicio Web de MusicBrainz es una interfaz de acceso a esta base de datos y por tanto a toda información que se ha descrito.

La arquitectura de su servicio sigue los principios de diseño REST, a los que se hace referencia en el apartado de este capítulo dedicado a REST. Por lo que la interacción con este servicio Web se hace utilizando HTTP y se obtiene una respuesta en formato XML, con todas las ventajas que esto aporta.

3.4.2. ¿Cómo se consigue la información?

La URL para acceder a esta información es `http://musicbrainz.org/ws/1/` añadiendo cuál es la categoría de la información buscada. Estas son las opciones disponibles, según lo que aparece en el sitio de MusicBrainz en el apartado dedicado a la descripción de su servicio Web:

`http://musicbrainz.org/ws/1/artist/`	Todos los artistas que compartan el nombre por el que se realiza la búsqueda
`http://musicbrainz.org/ws/1/artist/MBID`	Un artista en concreto
`http://musicbrainz.org/ws/1/release-group/`	Colección de todos los grupos de discos
`http://musicbrainz.org/ws/1/release-group/MBID`	Grupo de discos concreto

http://musicbrainz.org/ws/1/release/	Colección de todos los discos
http://musicbrainz.org/ws/1/release/MBID	Disco concreto
http://musicbrainz.org/ws/1/track/	Colección de todas las canciones
http://musicbrainz.org/ws/1/track/MBID	Canción individual
http://musicbrainz.org/ws/1/label/	Colección de todas las etiquetas
http://musicbrainz.org/ws/1/label/MBID	Sello musical o compañía concreta
http://musicbrainz.org/ws/1/tag/	Folcsonomías
http://musicbrainz.org/ws/1/rating/	Valoraciones

Tabla 1: Tipo de llamadas al servicio Web MusicBrainz [8]

Como se puede observar, para cada categoría hay dos URLS de acceso. Esto es debido a que hay dos opciones de búsqueda, si se conoce el identificador que la entidad buscada tiene dentro de MusicBrainz se puede hacer una búsqueda directa a través de las URLS terminadas en MBID. Si por el contrario no se conoce, habría que hacer una búsqueda previa para conseguirlo, que se hace a través de las URLS sin MBID.

Una vez se conoce el MBID, la consulta a la URL correspondiente puede indicar otros parámetros para moldear la cantidad de información que se desea en la respuesta. Estos parámetros varían dependiendo del tipo de entidad y se pueden consultar en detalle en: http://musicbrainz.org/doc/XML_Web_Service#Introduction

3.4.3. ¿Cómo se devuelve la información?

Este servicio Web nos ofrece la respuesta en formato XML, lo que aporta ventajas a la hora de ser asimilada por el cliente que ha realizado la petición. Como se ha comentado en el apartado anterior, hay muchos tipos de llamadas dependiendo del tipo de entidad por el que se pregunte, incluso se ha visto que los datos almacenados por cada una de ellas varía también, esto va a llevar a que el contenido y estructura de las respuestas varíen dependiendo del tipo de entidad.

Dentro de una misma entidad, las respuestas también van a variar dependiendo de la información que se haya requerido en la petición. Al ver el formato de las peticiones se ha hablado de una serie de opciones para incluir o no determinada información, es decir para recoger el valor de determinados atributos de la entidad concreta. Por tanto el contenido de la respuestas, es decir los atributos que se van a mostrar, va a depender también de la información que se demande en el campo opciones.

Este es un ejemplo de respuesta básico, que es el resultado de realizar una búsqueda genérica, es decir se ha introducido el nombre de un disco y se quieren obtener todas las coincidencias, para obtener los identificadores de MusicBrainz (MBID) de los mismos. En posteriores

capítulos se tratarán más a fondo los posibles formatos de respuesta, haciendo hincapié en los relativos a las entidades que se ven afectadas por este proyecto.

```
<?xml version="1.0" encoding="UTF-8"?>
  <metadata xmlns="http://musicbrainz.org/ns/mmd-1.0#">
    <release-list count="3">
      <release id="cac64a87-42f9-4c1c-a5ef-1e6824e20678"/>
      <release id="b71926ec-5813-4fa4-9372-0c07154a07af"/>
      <release id="172ddda3-1837-4fd2-8d12-ddd1e70b4c57"/>
    </release-list>
</metadata>
```

Antes también se podía demandar que el servicio Web de MusicBrainz devolviera la información utilizando el formato RDF. Esto fue desarrollado antes de que establecieran las mejores prácticas para los servicios Web, por lo que no es coherente, conciso y es difícil de utilizar. Por ello no debe utilizarse para nuevos desarrollos.

3.5. *Tecnología utilizada*

En este apartado se describen brevemente las tecnologías que se han decidido utilizar para el desarrollo de este proyecto con el objetivo de dar al lector una visión interna del sistema.

En primer lugar, y tras haber analizado las opciones puestas a disposición por cada uno de los servicios Web externos que intervienen, se describe, sin entrar en detalles de muy bajo nivel, cuál es el comportamiento ideal de sistema y cómo se va a comunicar con estos servicios Web.

La comunicación con los servicios Web externos se hará a través de REST, dado que el servicio Web de MusicBrainz sólo acepta llamadas a través de REST y se encuentra entre una de las opciones de comunicación con Open Calais.

Para poder realizar estas llamadas REST se utilizará Ajax debido a su sencillez y a que el equipo de desarrollo se siente más cómodo con este lenguaje JavaScript y esto evita incurrir en costes de formación adicionales. Además Ajax permite realizar estas llamadas de forma asíncrona, con lo que el usuario no tiene que estar esperando a que la página se recargue n veces por cada ve que se tenga que realizar una de las múltiples llamadas a los servicios Web externos.

La respuesta de estas llamadas se va a obtener en XML para el caso de MusicBrainz, ya que es el formato obligado por este servicio, y en JSON para Open Calais. Más adelante se hablará del por qué de la elección de JSON como formato de la respuesta, ya que OpenCalais da varias opciones de formatos.

Otra de las ventajas de Ajax es que va a mantener el XML de las respuestas de estos servicios, para el caso de MusicBrainz, o el modo texto de las mismas, para la respuesta JSON de Open Calais.

Para analizar y procesar las respuestas JSON de OpenCalais se utilizará una función provista por JavaScript para este tipo de formato (de ahí la elección del mismo), y para procesar las de MusicBrainz se utilizará DOM ya que permite un fácil acceso a los elementos de un XML.

A continuación se tratan las tecnologías referenciadas y algunas más que surgen del uso de las mismas.

3.5.1. Ajax

Ajax se trata de una técnica para desarrollar aplicaciones Web de manera que estén dotadas de interactividad, lo que se consigue gracias a la asincronía que aporta Ajax.

Las aplicaciones desarrollas con esta técnica, que no es más que JavaScript asíncrono y XML, se ejecutan en el lado del cliente, es decir en el navegador que éste esté utilizando para visualizar la Web.

La asincronía permite que el usuario pueda seguir utilizando el navegador mientras éste en segundo plano estará manteniendo comunicación con el servidor. Gracias a esta comunicación en segundo plano, la página que está visualizando el usuario puede sufrir cambios sin necesidad de ser recargada Estas aplicaciones se ejecutan en el cliente, es decir, en el navegador de los usuarios mientras se mantiene la comunicación asíncrona con el servidor en segundo plano. De esta forma es posible realizar cambios sobre las páginas sin necesidad de recargarlas, es decir sin interrumpir en la visualización ni en el comportamiento de la página.

Volviendo al hecho de que Ajax es JavaScript, es el lenguaje en el que se realizan las llamadas asíncronas al servidor, utilizando para ello el objeto XMLHttpRequest. En la Ilustración 5 se muestra su funcionamiento.

Pero aquí surge un problema, la llamada limitación de cross-domain scripting, es decir la limitación de ejecutar scripts procedentes de dominios diferentes. Esta limitación surge de una característica de seguridad que desde el navegador previene a los scripts del lado del cliente de acceder al contenido de otros dominios que sean diferentes al que pertenecen. Por tanto los scripts que contienen las llamadas a los servicios Web externos no se van a ejecutar correctamente al intentar comunicar con los dominios de los servicios Web que son externos.

Existen varias soluciones para resolver esta limitación, por ejemplo para Internet Explorer bastaría con cambiar la configuración de seguridad para que fuera más permisivo, pero esta solución no valdría para Firefox. Otras soluciones serían por ejemplo que los scripts estuvieran firmados por Autoridades de Certificación reconocidas, que se colocara un proxy inverso entre nuestro navegador y los servicios Web externos, o que se ejecutara desde el servidor el código en vez de ejecutarlo desde el lado del cliente, es decir, desde el navegador. Esta última solución es la más realista ya que valdría para todos los navegadores y es de sencilla implementación.

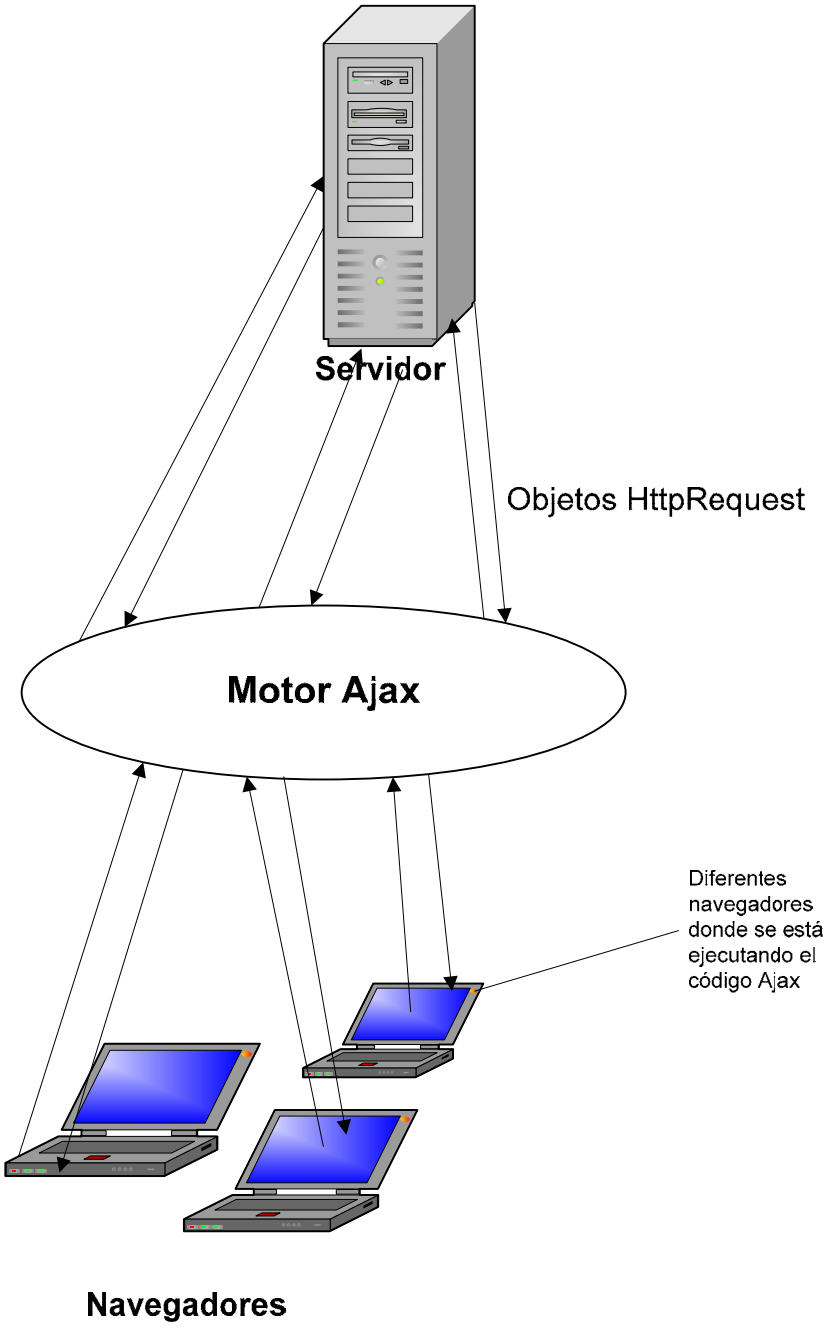


Ilustración 5: Funcionamiento de Ajax

3.5.2. PHP

PHP es un lenguaje de programación interpretado que se ejecuta en el lado del servidor, para el desarrollo de páginas Web dinámicas.

Como se ha dicho se ejecuta en el lado del servidor cogiendo el código PHP como entrada y creando a través del mismo páginas Web de salida. Una de sus ventajas es que se puede desplegar en la mayoría de servidores y sistemas operativos.

Al hablar de Ajax como solución para implementar el sistema, se encuentra el problema que presenta el llamado cross-domain scripting. De las posibles soluciones para el mismo, como se discutirá al tratar la arquitectura del sistema, la más sencilla es la utilización de PHP. En este sistema se va a utilizar PHP con la finalidad de que ejerza como proxy, a través de un pequeño fichero PHP se va a conseguir que este problema se solucione. PHP se encarga de realizar al servidor la llamada que el objeto XMLHttpRequest le pase y de recoger la llamada del servidor, para simplemente mostrarla.

3.5.3. DOM

DOM es una interfaz de programación (API) que proporciona una serie de objetos y métodos que permiten el acceso y la modificación de los elementos de documentos HTML y XML. Esto permite cambiar de manera dinámica el contenido de los documentos.

Para el sistema que nos ocupa su uso va a resultar práctico en dos momentos. El primero a la hora de acceder a los elementos de la página HTML a través de la cual se establece la comunicación con el usuario. Ya que se desea que esta página esté compuesta por contenido dinámico.

El segundo uso de esta interfaz es que, al permitir el acceso a los elementos de un XML, va a hacer posible realizar el análisis de las respuestas que se van a obtener del servicio Web de MusicBrainz. Por tanto toda la información disponible en las repuestas de MusicBrainz se va a extraer a través de los objetos y métodos de DOM para que puedan ser procesada según convenga en cada momento, ya que, como se ha visto con anterioridad, las respuestas van a variar dependiendo del tipo de llamada y de la entidad sobre la que se haga.

3.5.4. Apache

Apache es un servidor web de código abierto que funciona sobre múltiples plataformas, como Windows, Unix o Macintosh y que implementa el protocolo HTTP.

La incorporación de código PHP, que, si recordamos, se trata de código que se ejecuta en el servidor, hace necesaria la inclusión de un servidor Web en el sistema. Al ser Apache multiplataforma, de código abierto y uso sencillo se ha considerado como la mejor opción para este sistema.

3.5.5. JSON

Se trata de un formato ligero de intercambio de datos que debido a su simplicidad y se utiliza como alternativa al uso de XML.

Esta simplicidad es debida a que si se utiliza con JavaScript se dispone de un procedimiento llamado eval (), que se trata de una analizador semántico específico para JSON.

Por ello se eligió como formato de devolución del servicio Web de Calais, ya que al utilizar como lenguaje de programación JavaScript, el análisis de la respuesta es automático utilizando eval(), solo es necesario procesar la respuesta que este procedimiento da. Este procesamiento es muy sencillo ya que las entidades ya están agrupadas en categorías después de la ejecución de eval (), basta con recorrer cada uno de los elementos reconocidos como arrays.

4. Descripción del sistema

En este apartado se realiza una descripción del sistema, en primer lugar presentando los objetivos funcionales que se perseguían. Luego se aporta otra información más descriptiva como la arquitectura que sigue, diagramas de secuencia, de componentes, cómo realiza el acceso a los servicios Web externos, las librerías que utiliza y otra información que resulta útil para comprender desde dentro el funcionamiento del sistema.

El objetivo de este proyecto es aprovechar la información que determinados Servicios Web ponen a nuestra disposición para construir una herramienta que sea capaz de distinguir dentro de un texto determinadas entidades del mundo de la música y ampliar información relativa a las mismas.

Los Servicios Web de los que se va a extraer esta información son en concreto OpenCalais Web Service y MusicBrainz Web Service, de los que se ha hablado en el apartado anterior.

En concreto OpenCalais se va a encargar de realizar el correspondiente análisis semántico del texto y MusicBrainz de ampliar la información de las entidades que se hayan reconocido.

Por lo tanto se distinguen dos objetivos básicos, que se desarrollarán a continuación:

- Reconocimiento semántico.

- Ampliación de la información.

Reconocimiento semántico

La función del reconocimiento semántico, como ya se ha comentado, se va a llevar a cabo con la ayuda de la información que nos devuelve el servicio web de Open Calais.

La idea es que el usuario indique al sistema qué texto quiere reconocer, el sistema por su parte recogerá este texto y realizará una llamada a Open Calais, que devolverá a su vez el texto una vez ha realizado el reconocimiento semántico. Es ahora el sistema el que tiene que procesar esta respuesta para mostrarle al usuario la información de manera inteligible.

OpenCalais devuelve el texto indicando las entidades que ha conseguido analizar, es decir aquellos términos que ha conseguido reconocer y desambiguar en el caso de que haya sido necesario. Estas entidades se devuelven categorizadas dentro del grupo semántico al que pertenezcan.

Dado que el ámbito del sistema se limita al mundo de la música, de toda la información que Calais devuelva sólo será necesario mostrar la relacionada con este tema. En concreto las categorías de entidades que se van a ser reconocidas por el sistema son: Artistas, Grupos Musicales y Álbumes Musicales.

Por lo tanto el objetivo del reconocimiento semántico pasa por recoger un texto concreto, enviárselo a OpenCalais para que lo analice semánticamente y, cuando se haya obtenido la respuesta, procesarla para escoger sólo la información necesaria para este sistema.

Ampliación de la información

Como se indicado con anterioridad, el objetivo de la ampliación de la información va a ser llevado a cabo con la ayuda del servicio web de MusicBrainz.

En este caso, una vez se haya realizado el procesamiento de la información de manera semántica, el usuario debe tener la oportunidad de poder ampliar la información a cerca de cualquiera de las entidades que hayan sido reconocidas.

Información que hay que mostrar de cada uno de las categorías de entidades:

- Artista:
    - Si tiene o ha tenido hijos artistas, y la posibilidad de recopilar información de los mismos.
    - Si tiene o ha tenido matrimonios con otros artistas, y la posibilidad de recopilar información de los mismos.
    - Si ha pertenecido o pertenece a uno o varios grupos musicales, y la posibilidad de recopilar información acerca de los mismos.
    - Información de los sitios Web donde pueda seguir recogiendo información del propio artista. Como por ejemplo enlace a Wikipedia, BBCMusic, Club de Fans, etc.
    - Valoración que otros usuarios hacen del mismo.

- Grupo Musical:
    - Componentes que forman o han formado parte del mismo, y la posibilidad de ampliar información de cada uno de ellos.
    - Álbumes que han sacado al mercado, y la posibilidad de ampliar información de cada uno de ellos.
    - Información de los sitios Web donde pueda seguir recogiendo información del propio grupo. Como por ejemplo enlace a Wikipedia, BBCMusic, Club de Fans, etc.
    - Valoración que otros usuarios hacen del mismo.

- Álbum Musical:
    - Versiones que han salido al mercado indicando su formato y fecha de salida.
    - Detalle de las pistas que forman parte del álbum, indicando su título, la duración y dando la posibilidad de ampliar información acerca de las mismas.
    - Información de los sitios Web donde pueda seguir recogiendo información del álbum e incluso enlaces a sitios dónde se pueda comprar. Como por ejemplo enlace a Wikipedia, BBCMusic, Amazon, etc.
    - Valoración que otros usuarios hacen del mismo.

Toda esta información tiene que ser mostrada al usuario de manera ordenada, y siempre teniendo en cuenta que puede ser que se carezca de determinada información por lo tanto debe ser consistente con lo obtenido del servicio web MusicBrainz.

4.1. *Entorno de desarrollo*

Este apartado describe el entorno de desarrollo con el que se ha trabajo durante la construcción del sistema.

Como entorno de desarrollo se puede utilizar cualquier editor que reconozca los siguientes lenguajes: JavaScript, PHP, HTML y XML, que son los que van a intervenir durante el desarrollo del sistema. En concreto se ha utilizado el editor Notepad++.

Al tratarse con lenguajes que no necesitan de un compilador no hace falta utilizar entornos de desarrollo más sofisticados como puede ser Eclipse, JavaScript se va a ejecutar en el navegador y PHP en el servidor.

Para poder ir comprobando el contenido de las respuestas HTTP de cada uno de servicios, así como las peticiones a cada uno de los mismos se ha utilizado la extensión del navegador Mozilla Firefox Bugzilla. Con ella se puede ver el estado de cada petición además de las respuestas detalladas de cada llamada, también se detalla el tiempo empleado en cada llamada desde que se inicia hasta que se obtiene la respuesta.

4.2. *Arquitectura*

Este apartado muestra la arquitectura que sigue el sistema, además de realizar una comparativa con las otras opciones que se podían haber utilizado.

La solución propuesta para desarrollar el sistema cumpliendo con los objetivos funcionales comentados al inicio de este capítulo, pasa por realizar dos llamadas diferentes, unas al servicio Web de OpenCalais y otras al de MusicBrainz, y procesar la información que se haya recibido.

Hay dos opciones en cuanto al lugar dónde ejecutar el código, utilizar un lenguaje del lado del servidor o utilizar un lenguaje del lado del cliente, es decir en el navegador. Se ha optado por esta última aunque las dos opciones tienen sus ventajas e inconvenientes. Por ejemplo los lenguajes del lado del cliente son totalmente independientes del servidor con lo que la página podrá estar albergada en cualquier servidor, mientras que éstos últimos tienen la ventaja de que no dependen del navegador que se esté utilizando ni de la versión del mismo.

A continuación se muestran las dos opciones ilustradas gráficamente:

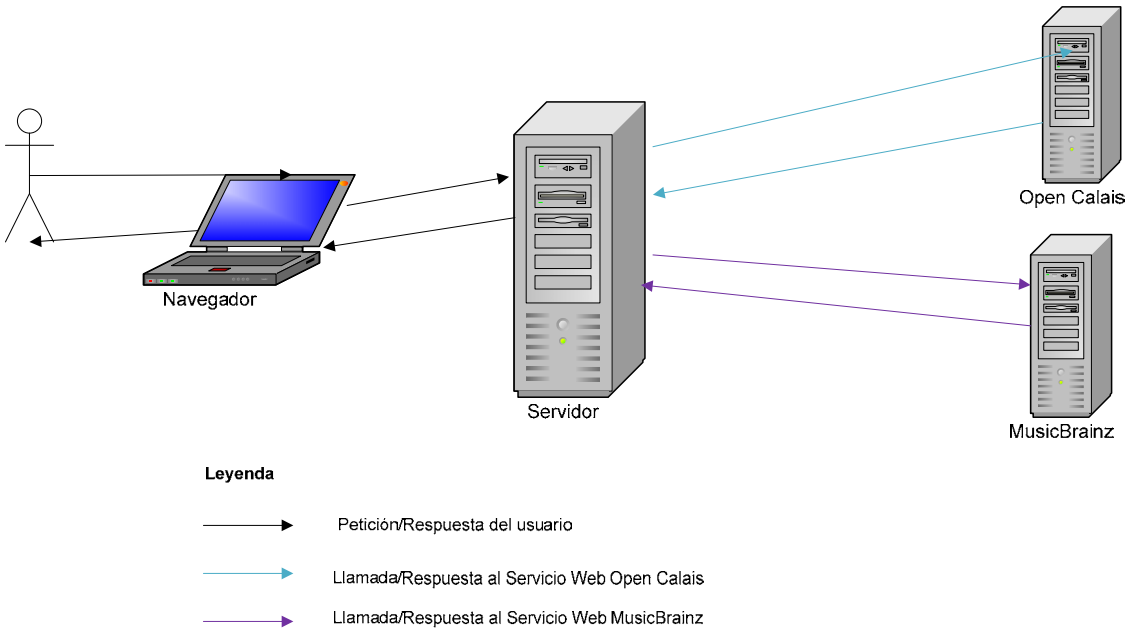


Ilustración 6: ***Ejecución del código del lado del servidor***

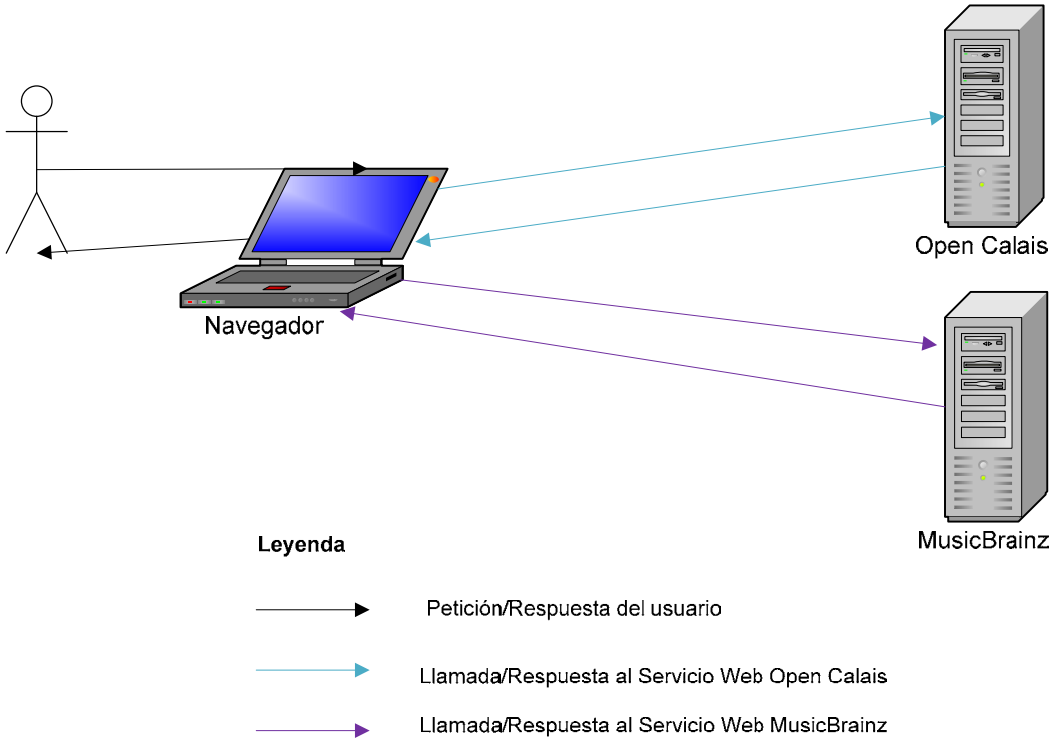


Ilustración 7: ***Figura Ejecución del código del lado del cliente***

Finalmente se ha optado por la opción de ejecutarlo en el lado del servidor por no depender de un servidor donde alojar el código y por tener el equipo de desarrollo más conocimiento de los lenguajes del lado del cliente que de los del servidor.

La arquitectura final no queda tan sencilla como la propuesta en la Ilustración 7, ya que hay una restricción de seguridad en los navegadores que impiden que se ejecuten scripts procedentes de sitios que no estén dentro del dominio propio. Esta restricción se aplica en lo navegadores para evitar un agujero de seguridad que es conocido como Cross-site sripting. Este problema se da ya que no se pueden ejecutar scripts de un sitio Web que no se encuentre dentro de su mismo dominio.

Esta restricción, para Internet Explorer se soluciona bajando el nivel de seguridad, pero esto puede traer consecuencias negativas al realizar el resto de visitas a otros sitios Web dejando en peligro el navegador del usuario. Para Mozilla Firefox ni siquiera se puede eliminar esta restricción cambiando la configuración del mismo.

Una de las soluciones para poder realizar este tipo de peticiones evitando esta restricción es hacer las llamadas a través de un proxy, que va a estar formado por un código PHP que es código que se ejecuta en el servidor. El código PHP va a estar alojado en un servidor Apache.

A continuación se muestra un gráfico que intenta explicar las distintas etapas de comunicación que se establecen entre el sistema y el los servicios Web externos que van a intervenir para alcanzar la solución.

El sistema está formado por el navegador y el servidor Apache. Esto es debido a que parte del código se ejecutará en el lado del cliente, es decir en el navegador, y otra parte tendrá que ser ejecutado en un servidor Web, en este caso un servidor Apache.

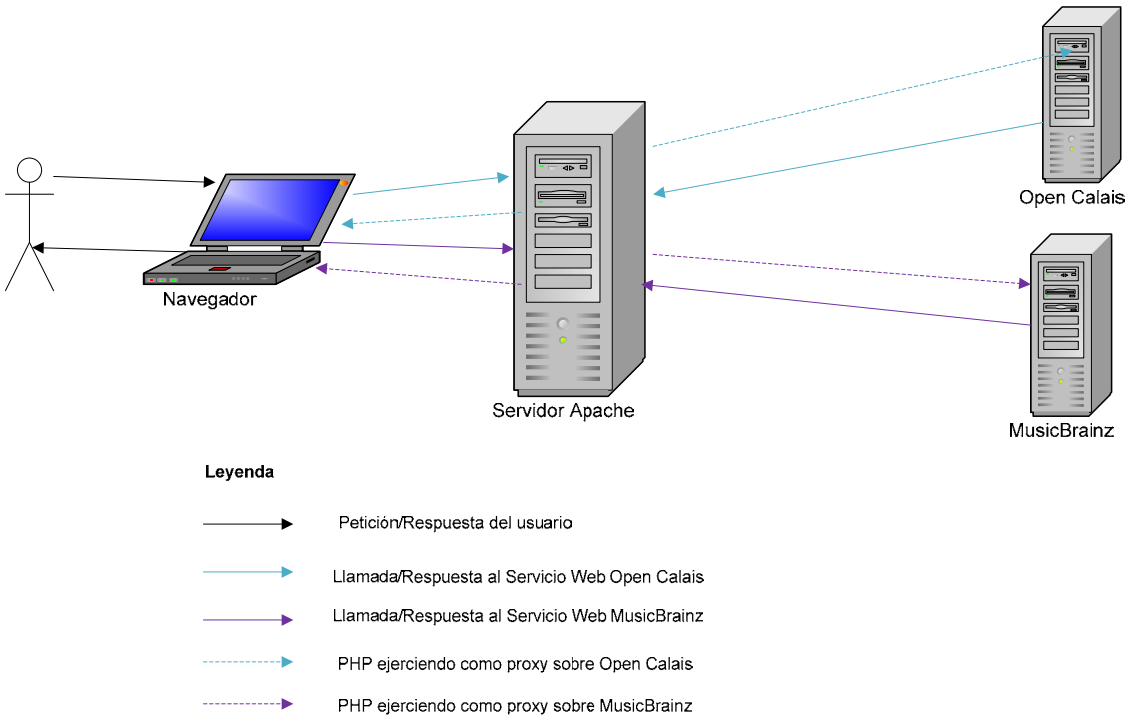


Ilustración 8: Arquitectura final del sistema

El usuario realizará una sola petición y el sistema se encargará de realizar las sucesivas peticiones. A la vista del diagrama, se puede observar que hay dos procesos de comunicación diferenciados que parten del navegador, la comunicación que establece el cliente con el servicio Web de OpenCalais y la que establece con el servicio Web de MusicBrainz.

El usuario realiza la petición a través del navegador Web, y es éste el que realiza la primera llamada a través del servidor Apache que actúa aquí como proxy. El servidor realizará la misma petición al servicio Web de Open Calais, que le devolverá una respuesta con el texto procesado. Esta respuesta el servidor se la devuelve al navegador, que, tras realizar las transformaciones pertinentes, realizará al servidor tantas llamadas como entidades haya reconocido Calais. Nuevamente el servidor que actuará como proxy pasando la petición a MusicBrainz, que le devolverá la respuesta, que el servidor transmitirá al navegador. Tras realizar nuevas transformaciones se le devuelve la información al usuario.

4.3. Diagramas de secuencia

En este apartado se analiza el uso del sistema y el tránsito de información que se produce durante el uso del mismo a través de una serie de diagramas de secuencia. Por cada una de las operaciones que se pueden realizar en el sistema, se mostrará el tránsito primero en el caso ideal y después, en el caso de que pueda darse, en caso de error.

Este sistema permite realizar dos operaciones, Analizar y Borrar. La funcionalidad más importante recae en la operación de Analizar, ya que el borrado es simplemente el restablecimiento del cuadro de texto original sin ningún contenido. Por tanto se va a analizar únicamente la operación de Analizar.

4.3.1. Operación de Analizar sin error:

Al llegar a la página principal del sistema el usuario introducirá el texto que desee reconocer semánticamente y acto seguido pulsará el botón Analizar correspondiente. El sistema en ese punto recoge el texto introducido y se lo envía al servicio Web de OpenCalais para que comience el reconocimiento semántico. OpenCalais devolverá un fichero JSON que contiene todas las entidades que ha sido capaz de reconocer, junto con sus categorías y, además de otra información que no es necesaria para este sistema, las coordenadas que ocupan dentro del texto. El sistema realizará entonces el procesado de esta respuesta analizando qué entidades pertenecen al ámbito musical y en concreto a las categorías propias de este sistema.

Cada entidad que ha reconocido la transforma en un enlace HTML, que al ser pulsado realiza la llamada a MusicBrainz para ampliar la información. Por tanto en este punto el usuario ya ha recibido una respuesta y puede observar que se han sustituido las entidades musicales en el texto que había introducido inicialmente.

Mientras el usuario ya tiene a la vista el texto final, el sistema está realizando forma síncrona todas las llamadas a MusicBrainz para que el usuario disponga ya de esta información

procesada cada vez que pulse una entidad y no tenga que esperar a que se realice la llamada cuando piche sobre la entidad y además tenga que esperar a que se procese la respuesta.

En este punto, por cada entidad el sistema realizará dos llamadas, la primera como búsqueda para obtener el identificador de la entidad dentro de la base de datos de MusicBrainz, y la segunda través de este identificador y obteniendo más información de la entidad. Una vez obtenga la información de la entidad la procesará para mostrar la necesaria y de la manera estipulada. Se creará entonces un tooltip para cada entidad del texto rellenándolo con esta información procesada y maquetada.

A continuación se muestra el diagrama de secuencia propio de esta acción:

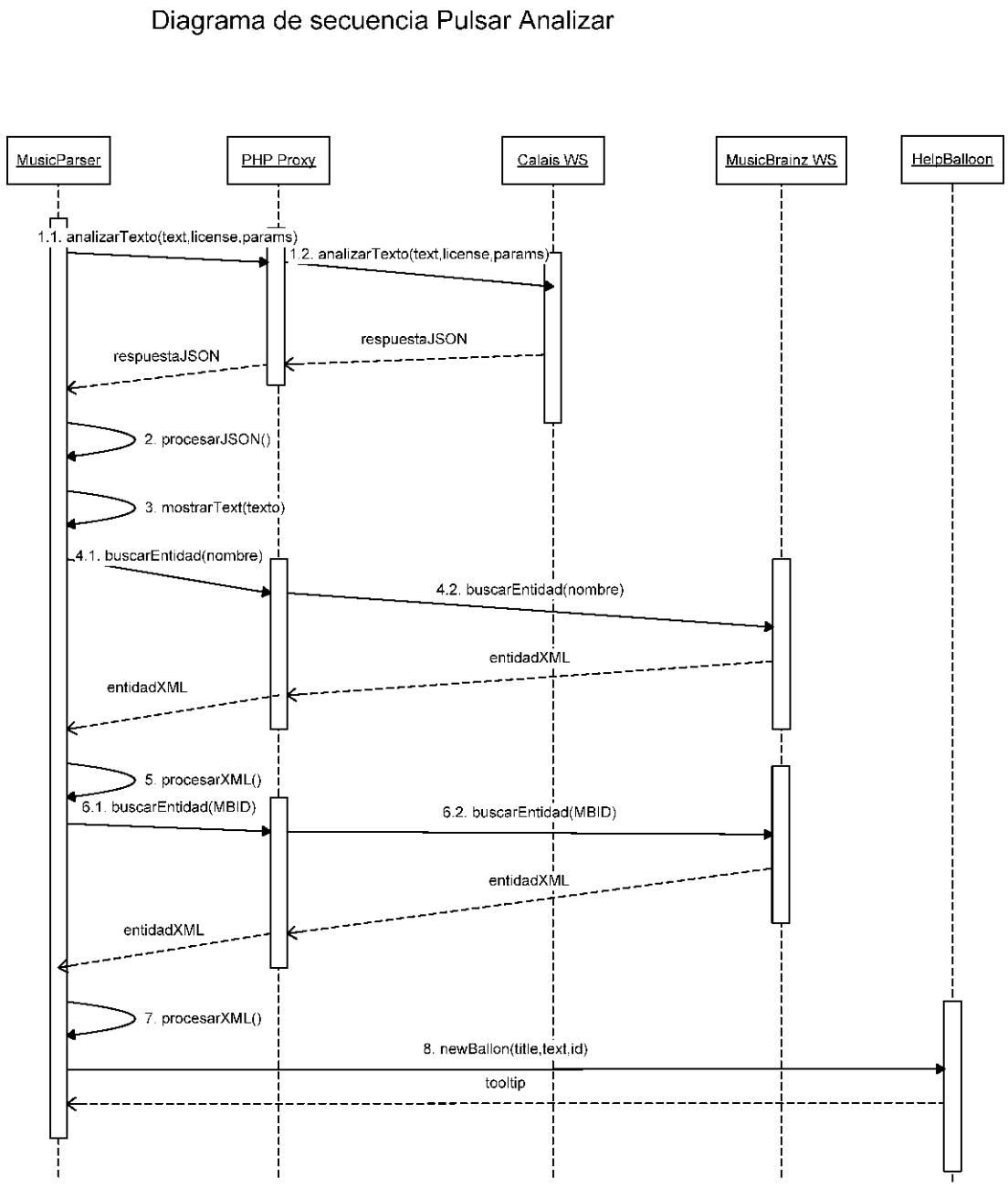


Ilustración 9: Diagrama de secuencia Pulsar Analizar

Cuando el usuario pulsa Analizar esto es lo que ocurre interiormente en el sistema:

1.1. **analizarTexto (texto, license, params):** lo primero que hace el sistema es coger el texto que el usuario ha introducido, el número de licencia y los parámetros de configuración de Calais de que dispone con anterioridad, y hace una llamada al servicio Web de Calais con ellos. Esta llamada se hace a través del proxy, por tanto le hace el envío al mismo.

1.2. **analizarTexto (texto, license, params):** la misma información que le ha llegado al proxy PHP es la que transmite al servicio Web de Calais. El proxy espera a que le llegue la respuesta en formato JSON (es lo que se ha indicado en los parámetros a Calais) para devolvérsela a MusicParser.

2. **procesarJSON():** se procesa la respuesta JSON recibida de Calais, seleccionando de entre todas las entidades reconocidas las que se necesitan para este sistema, es decir los Artistas, Grupos Musicales y Álbumes Musicales.

3. **mostrarTexto (texto):** se muestra el texto procesado, aunque no esté todavía finalizado el procesamiento por parte del sistema.

   Los pasos que se muestran a continuación, desde el 4 al 8, se van a repetir para cada una de las entidades de contexto musical que se hayan reconocido por Calais:

4.1. **buscarEntidad (nombre):** se hace una llamada a MusicBrainz a través del proxy indicándole el nombre de la entidad de la que se desea ampliar la información.

4.2. **buscarEntidad (nombre):** el proxy redirige la llamada a MusicBrainz tal y como le ha llegado y espera a que le llegue la respuesta en forma de XML con todas las coincidencias que MusicBrainz ha encontrado con ese nombre.

5. **procesarXML ():** se procesa la respuesta XML que ha llegado para extraer el identificador o MBID de la entidad.

6.1. **buscarEntidad (MBID):** se realiza otra llamada a través del proxy, esta vez preguntando por una entidad concreta a través del MBID.

6.2. **buscarEntidad (MBID):** se redirige la llamada desde el proxy a MusicBrainz y se espera a que devuelva el resultado en forma de XML.

7. **procesarXML:** se procesa el XML de la repuesta para que se muestre sólo la información deseada y de la manera deseada dentro del tooltip.

8. **newHelpBallon:** se crea un nuevo tooltip haciendo una llamada a la librería HelpBalloon, a la que se le indica el título que va a llevar el tooltip, el contenido del mismo que es la información extra que se ha obtenido de la entidad y un identificador para identificar dentro del texto el tooltip.

Una vez hecho esto para cada una de las entidades reconocidas, y si no ha ocurrido ningún error durante, al pulsar sobre una entidad se verá la información obtenida de la misma de manera resumida.

4.3.2. Operación Analizar con error en Calais:

Para la operación anterior, el movimiento interno del sistema varía si al realizar la primera llamada con el texto que el usuario ha introducido el servicio Web de OpenCalais devuelve un error, ya sea porque ha ocurrido un error de procesamiento y OpenCalais devuelva un texto que no esté en formato JSON correctamente o porque no se haya podido establecer la conexión con el servicio Web sea cual sea la causa. Al procesar la respuesta y ésta no coincidir con el formato esperado aparecerá una ventana de error con el mensaje correspondiente.

A continuación se muestra el diagrama de secuencia para este caso:

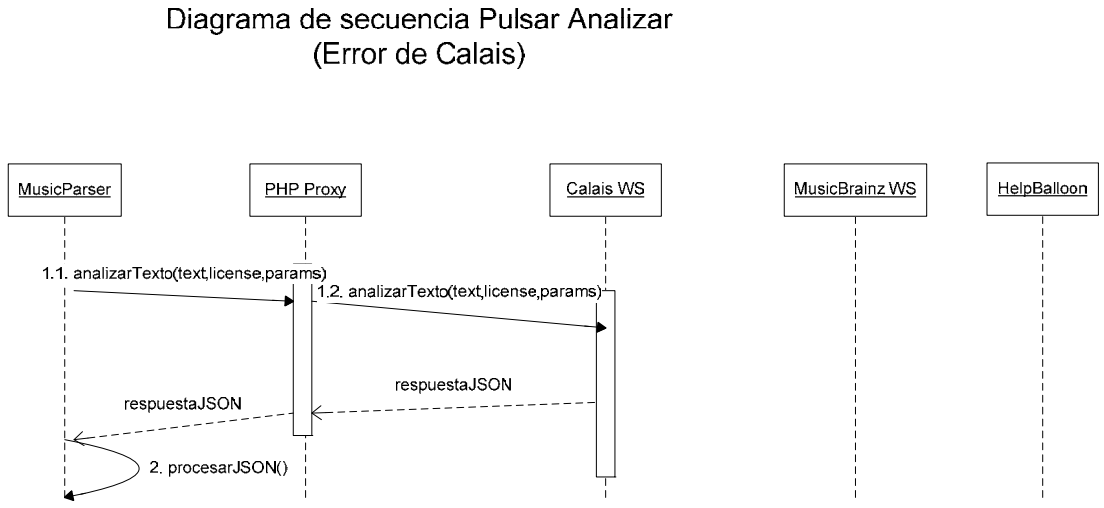


Ilustración 10: Diagrama de secuencia Pulsar Analizar con Error

1.1. analizarTexto (texto, license, params): lo primero que hace el sistema es coger el texto que el usuario ha introducido, el número de licencia y los parámetros de configuración de Calais de que dispone con anterioridad, y hace una llamada al servicio Web de Calais con ellos. Esta llamada se hace a través del proxy, por tanto le hace el envío al mismo.

1.2. analizarTexto (texto, license, params): la misma información que le ha llegado al proxy PHP es la que transmite al servicio Web de Calais. El proxy espera a que le llegue la respuesta en formato JSON (es lo que se ha indicado en los parámetros a Calais) para devolvérsela a MusicParser. En este caso ha ocurrido un error interno de Calais que no ha podido analizar el texto, así que devuelve un XML con error.

2. procesarJSON (): se procesa la respuesta JSON recibida de Calais, que en este caso es un mensaje de error. Como el método espera recibir un JSON ocurre un error de procesamiento y se le avisa al usuario del mismo.

En este caso el error ha sido de Calais, pero el funcionamiento en caso de que no se llegara a establecer conexión con el mismo, el error mostrado al usuario sería el mismo.

4.3.3. Operación Pulsar sobre entidad reconocida:

Una vez se ha procesado todo el texto y, si no ha ocurrido ningún error durante ese procesamiento, el usuario podrá pulsar dentro del texto sobre la entidad reconocida que desee. Podrá saber cuáles se han reconocido porque se muestran como enlaces HTML.

Una vez se han procesado todas las entidades con la información que se ha obtenido de MusicBrainz, si el usuario pulsa sobre cualquiera de ellas, aparecerá un tooltip con la información de la entidad.

Este es el diagrama de secuencia, que es muy sencillo porque todo el procesado y construcción de la respuesta se ha realizado durante la etapa del Análisis anterior:

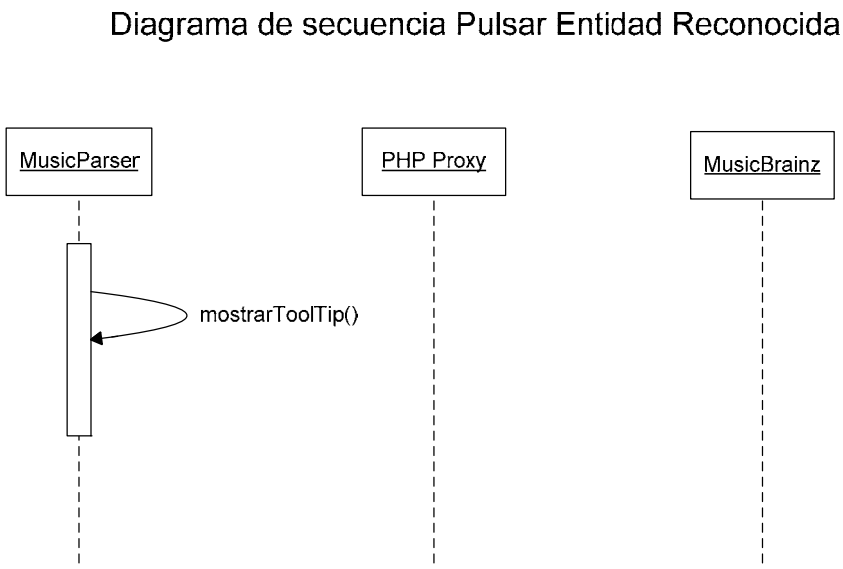


Ilustración 11: Diagrama de secuencia Pulsar sobre entidad reconocida

Este diagrama de secuencia muestra lo que ocurre cuando, una vez procesado el texto, el usuario pulsa alguno de las entidades que aparecen representadas en el texto como enlace.

Este es el proceso:

mostrarTooltip: como ya se ha procesado toda la información de las entidades, al pulsar sobre cualquiera de ellas aparece la información recopilada sobre la misma.

4.3.4. Operación Pulsar sobre entidad reconocida con error de MusicBrainz:

Al describir la operación de Analizar, se comentó que las llamadas a MusicBrainz y el procesamiento necesario de las respuestas que éste devuelve, se realizaban de manera síncrona mientras el usuario ya podía observar el texto reconocido. El usuario al pulsar sobre una entidad no tendría que esperar a que se realizaran las llamadas y el procesamiento correspondiente, ya que todo este trabajo ya se habría realizado.

Este es el caso ideal, de que no se haya producido ningún error durante el procesamiento o la obtención de la respuesta, ya que si se ha producido, el sistema habrá abortado este procesamiento masivo de la entidad que haya fallado y de las que no haya procesado hasta

ese momento. Por tanto en el momento en que el usuario vaya a pulsar sobre esta entidad, el sistema todavía no tendrá la información correspondiente a la misma.

La solución es incorporar las mismas llamadas que se han hecho durante el procesado, pero esta vez de manera asíncrona y cuando el usuario haya demandado esta información. Es decir, si el sistema no dispone de información de una entidad en el momento de que el usuario ha pulsado sobre la misma, el sistema realiza la primera llamada a MusicBrainz para conseguir el identificador de la entidad y luego la segunda para conseguir la información de la entidad de manera asíncrona sin interrumpir la interacción del usuario con el sistema y no teniendo que recargar la página.

A continuación se muestra el diagrama de secuencia correspondiente a este caso:

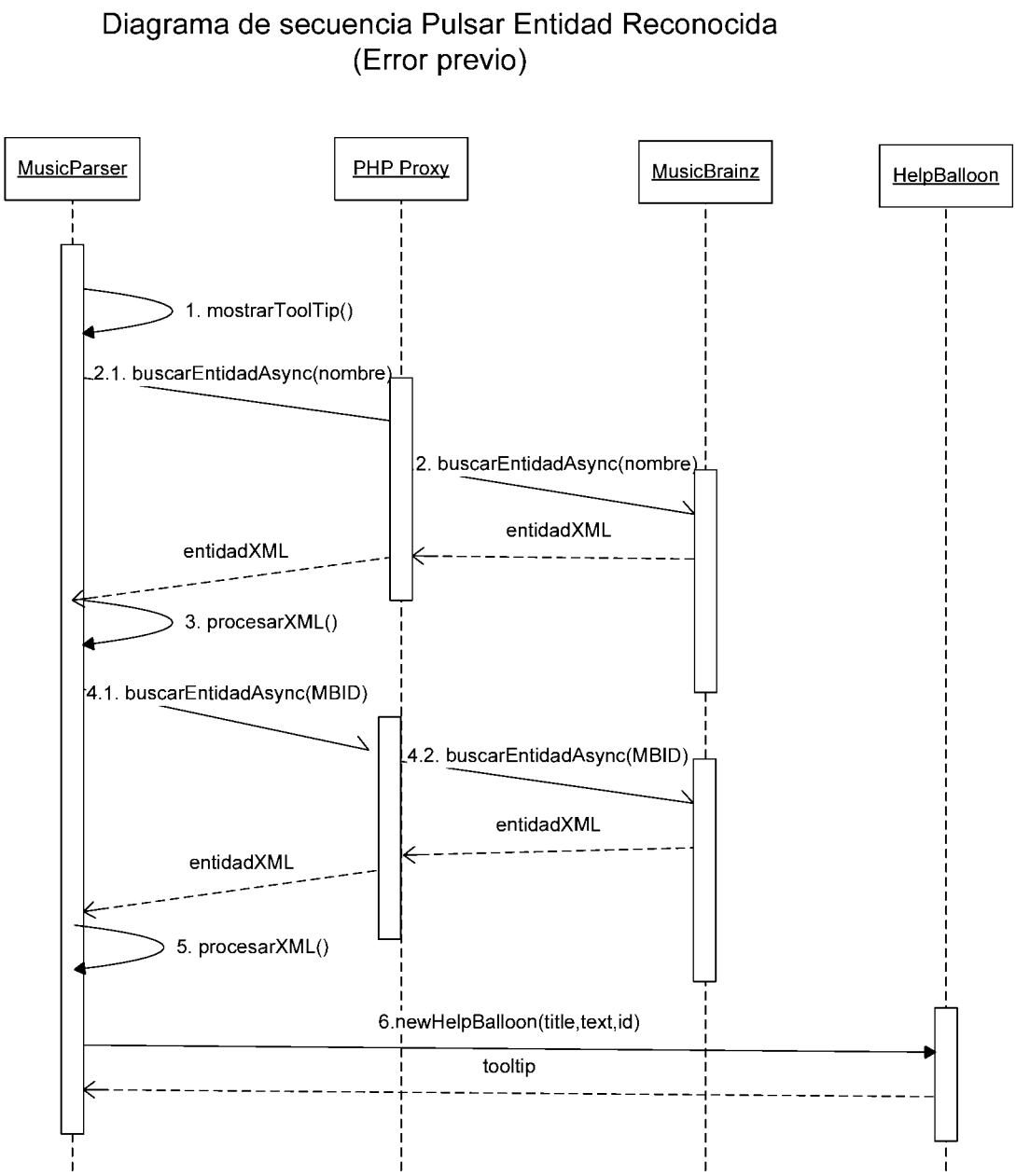


Ilustración 12: Diagrama de secuencia Pulsar sobre entidad reconocida con Error

Este diagrama trata el caso de que se haya producido un error de comunicación con MusicBrainz durante el proceso de Análisis. No se habrá interrumpido el mismo pero si se pulsa sobre una entidad de la que no se haya obtenido la información este el funcionamiento:

1. **mostrarTooltip:** se ha procesado toda la información de las entidades y al pulsar sobre cualquiera de ellas aparece la información recopilada sobre la misma pero en el caso de que se trate de una entidad de la que no se haya podido obtener información por algún error de procesamiento durante el proceso de análisis, se llama a los métodos asíncronos.

2.1. buscarEntidadAsync (nombre): se hace una llamada a MusicBrainz a través del proxy indicándole el nombre de la entidad de la que se desea ampliar la información. Esta vez la llamada se hace de manera asíncrona para que el usuario no tenga la necesidad de recargar ni deje de observar la página tal y como ha quedado con la información procesada.

2.2. buscarEntidad (nombre): el proxy redirige la llamada a MusicBrainz tal y como le ha llegado y espera a que le llegue la respuesta en forma de XML con todas las coincidencias que MusicBrainz ha encontrado con ese nombre.

3. **procesarXML ():** se procesa la respuesta XML que ha llegado para extraer el identificador o MBID de la entidad.

4.1 buscarEntidad (MBID): se realiza otra llamada a través del proxy, esta vez preguntando por una entidad concreta a través del MBID. Esta llamada también se realizara de manera asíncrona para que quede transparente al usuario.

4.2 buscarEntidad (MBID): se redirige la llamada desde el proxy a MusicBrainz y se espera a que devuelva el resultado en forma de XML.

5. **procesarXML:** se procesa el XML de la repuesta para que se muestre sólo la información deseada y de la manera deseada dentro del tooltip.

6. **newHelpBallon:** se crea un nuevo tooltip haciendo una llamada a la librería HelpBalloon, a la que se le indica el título que va a llevar el tooltip, el contenido del mismo que es la información extra que se ha obtenido de la entidad y un identificador para identificar dentro del texto el tooltip.

Este proceso tendrá lugar cuando ocurra algún error inesperado, ya que si se trata de errores que devuelva MusicBrainz el sistema está preparado para repararlos durante el procedimiento del Análisis.

4.4. *Interfaz de usuario*

La interfaz tiene un diseño muy sencillo. Lo único que es necesario es un cuadro de texto para que el usuario pueda introducir el texto que desee analizar y un par de botones. Es necesario que uno de los botones invoque al análisis del texto y otro de ellos que realice el borrado del cuadro de texto para que se pueda utilizar más de una vez.

Una vez se pulse el botón de Analizar, el cuadro de texto va a desaparecer para mostrar el texto analizado con las entidades reconocidas en forma de enlace.

Al pulsar el botón Limpiar volverá a aparecer el cuadro de texto vacío.

Dado que el procesamiento del texto va a tardar cierto tiempo es conveniente que se desactive el botón de Analizar hasta que se vuelva a pulsar el de borrado y aparezca de nuevo el cuadro de texto.

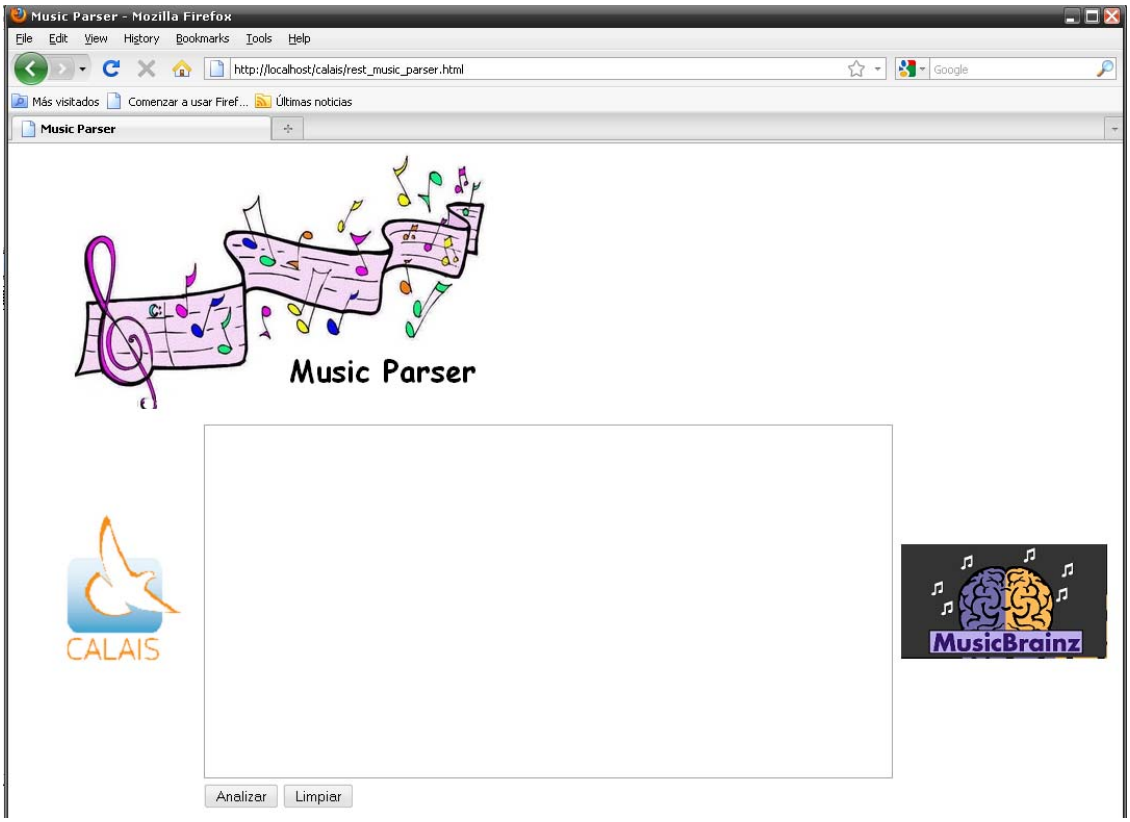


Ilustración 13: Interfaz de usuario del sistema

4.5. *Descomposición por componentes*

En este apartado se muestra la descomposición del sistema en componentes representando esta descomposición a través de un diagrama de componentes. Según la definición de Wikipedia: "Un diagrama de componentes representa cómo un sistema de software es dividido en componentes y muestra las dependencias entre estos componentes. Los componentes físicos incluyen archivos, cabeceras, bibliotecas compartidas, módulos, ejecutables, o paquetes." [7]

La descomposición por componentes se ha hecho teniendo en cuenta las funciones que cumple cada uno de los archivos con código JavaScript.

A continuación se muestra el diagrama de componentes del sistema con su correspondiente descripción:

Diagrama de componentes

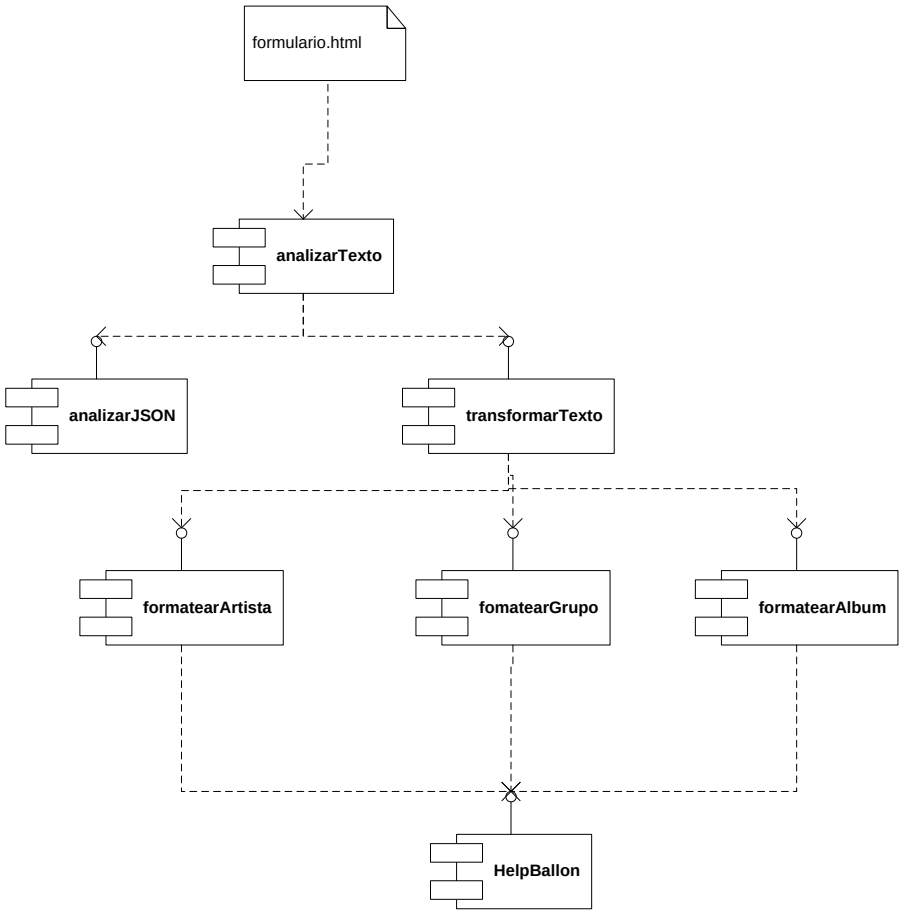


Ilustración 14: Diagrama de componentes del sistema

analizarTexto:

Este componente se encarga de, una vez lanzado el evento por el formulario, gestionar la llamada al servicio Web de Calais con los argumentos necesarios. Además se encarga de gestionar la transformación del texto introducido en el formulario con la ayuda de los componentes analizarJSON y transformarTexto.

analizarJSON:

Este componente presta su funcionalidad al componente analizarTexto, ya que éste necesita que se analice la respuesta que le haya devuelto el servicio Web Calais en formato JSON. Por ello analizarJSON analiza la jerarquía de esta respuesta extrayendo las diferentes entidades reconocidas de la misma y devolviéndole a analizarTexto un resumen de la respuesta JSON únicamente con los datos que le interesan, como son el nombre de la entidad, su categoría (si es Artista, Grupo o Álbum) y las coordenadas del texto donde aparece.

transfomarTexto:

Este componente se encarga de marcar en el texto las entidades que el componente analizarTexto le encomiende. Para sustituir en el texto correctamente cada una de las entidades con su correspondiente enlace o ampliación de la información necesita de la ayuda de los componentes formatearArtista, formatearGrupo y formatearAlbum. Por cada entidad que analizarTexto haya considerado como tal tiene que realizar una transformación que variará dependiendo de la categoría de la misma, por lo que recogerá la ayuda del componente que se corresponda.

formatearArtista:

Este componente ayuda transformarTexto a realizar las llamadas al servicio Web MusicBrainz para conseguir la infamación convenida de un artista determinado. Realizará al servicio Web tantas llamadas como sean necesarias hasta obtener la información que se desea mostrar al usuario final.

También se encarga de formatear esta información para hacerla presentable, para lo que necesita la ayuda del componente externo HelpBalloon.

formatearGrupo:

Este componente ayuda transformarTexto a realizar las llamadas al servicio Web MusicBrainz para conseguir la infamación convenida de un grupo musical determinado. Realizará al servicio

Web tantas llamadas como sean necesarias hasta obtener la información que se desea mostrar al usuario final.

También se encarga de formatear esta información para hacerla presentable, para lo que necesita la ayuda del componente externo HelpBalloon.

formatearAlbum:

Este componente ayuda transformarTexto a realizar las llamadas al servicio Web MusicBrainz para conseguir la infamación convenida de un álbum musical determinado. Realizará al servicio Web tantas llamadas como sean necesarias hasta obtener la información que se desea mostrar al usuario final.

También se encarga de formatear esta información para hacerla presentable, para lo que necesita la ayuda del componente externo HelpBalloon.

HelpBalloon:

Se trata de un componente externo, ya que es una librería, que va ayudar a los componentes formatearArtista, formatearGrupo y formatearAlbum a presentar la información que ellos formateen al usuario final. Éste les prestará un tooltip, o pequeña ventana de ayuda, que será rellenado con la información que convenga en cada momento y que estará asociado a una entidad concreta.

4.6. *OpenCalais Web Services*

Siguiendo las pautas que se describen en el apartado dedicado a este servicio Web (Ver apartado 3.3), hay tres puntos importantes a definir antes de comenzar el desarrollo:

La configuración que se desea de Calais, esto se hace a través de una serie de parámetros configurables que deben serle indicados a Calais.

Cómo se van a hacer las llamadas, ya se comentó que había varias maneras de conseguir la información, haciendo la llamada a través de SOAP, REST o Http Traffic Compression.

En qué formato se requiere que sea devuelta la información, también se vio que hay varios formatos de respuesta, RDF, Texto simple, Microformatos o JSON.

Para el desarrollo del sistema se ha seleccionado como método de realizar la llamada REST, y se desea que el formato de devolución sea JSON. A continuación se defenderán como solución óptima los formatos seleccionados sobre los que no lo han sido.

4.6.1. Configuración

Los parámetros de configuración los consume el servicio Web en formato XML. Hay una serie de ellos que son de uso obligatorio. Se puede consultar la Web oficial la tabla en la que aparecen los parámetros configurables y las opciones disponibles para cada uno de ellos.

Para el caso concreto que nos ocupa, necesitamos indicarle al servicio Web que la información se la vamos a pasar en formato texto, ya que es el usuario el que va a seleccionar el texto en claro que desea analizar, y que se desea que se devuelva la respuesta en formato JSON.

También necesitamos indicarle que tipo de metadatos necesitamos que extraiga el servicio, en este caso indicamos Generic Relations, que serían las que necesitamos al estar buscando entidades concretas, como son Artistas, Grupos musicales y Álbumes musicales. La otra opción es añadir Social Tags para reconocer términos más "relajados", que no es el caso, al tratarse de entidades fijas.

El resto de parámetros se dejarán con el valor por defecto, con el valor recomendado o sin indicar, ya que no son necesarios para la configuración elegida.

El XML de los parámetros quedaría:

```
<?xml version="1.0" encoding="UTF-8"?>
<c:params xmlns:c=\"http://s.opencalais.com/1/pred/\" xmlns:rdf=\"http://www.w3.org/1999/02/22-rdf-syntax-ns#\">
  <c:processingDirectives c:contentType=\"text/txt\" c:enableMetadataType=\"GenericRelations\"
  c:outputFormat=\"Application/JSON\" c:docRDFaccesible=\"true\" ></c:processingDirectives>
  <c:userDirectives c:allowDistribution=\"true\" c:allowSearch=\"true\" c:externalID=\"17cabs901\" c:submitter=\"ABC\"></c:userDirectives>
</c:params>
```

Ilustración 15:Ejemplo paso de parámetros en XML al servicio Web de OpenCalais

4.6.2. Llamadas

Las llamadas han decidido hacerse a través de llamadas REST siguiendo la operación POST, las razones que han llevado a tomar esta decisión son las siguientes:

- ✓ En general REST supone un consumo de recursos muy bajo.
- ✓ Las instancias del proceso son creadas explícitamente.
- ✓ El cliente no necesita información de enrutamiento a partir de la URI inicial.
- ✓ Los clientes pueden tener una interfaz genérica para las notificaciones.
- ✓ Es fácil de construir e integrar.

Y SOAP presenta las siguientes dificultades de implementación:

- ✓ Los clientes necesitan saber las operaciones y su semántica antes del uso.
- ✓ Los clientes necesitan puertos dedicados para diferentes tipos de notificaciones.
- ✓ Las instancias del proceso son creadas implícitamente.

Aunque el servicio Web de Calais ofrece la opción de realizar las llamadas REST a través de GET y de POST, se ha decidido optar por utilizar la operación POST ya que la cantidad de información que se puede transmitir haciendo uso de ella es mucho mayor que la que se podría transmitir con GET.

Calais necesita para dar respuesta a una petición que en su llamada se incluya el número de licencia, que se obtiene a través de realizar el registro en su página oficial, el texto que se desea analizar y los parámetros de configuración seleccionados, los comentados en el apartado anterior. La estructura de los argumentos que hay que pasarle al servicio Web debe ser así:

```
licenseID=url-encoded-string&content=url-encoded-string&paramsXML=url-encoded-string
```

Si la operación REST seleccionada hubiera sido GET estos parámetros viajaría en la URL, pero al ser POST esto parámetros deben ir el cuerpo de la petición http y debe tener como tipo de contenido (content type) ''application/x-www-form-urlencoded".

La URL a la que hacer la petición es: http://api.opencalais.com/enlighten/rest/

4.6.3. Respuestas

Para elegir el formato de la respuesta hay que tener en cuenta que ésta no es el final del flujo de datos, si no que tiene que ser procesado para que otro servicio, MusicBrainz, se encargue de ampliar la información pertinente. Por esta razón se desecharon los formatos RDF y microformatos, ya que son útiles si se dispone de un servicio que consuma datos en este formato y este no es el caso, ya que lo que se desea es realizar un análisis de la información recibida para seleccionar únicamente la de ámbito musical. Por lo tanto el formato preferible es JSON, ya que es de fácil análisis a través de JavaScript.

4.7. *MusicBrainz Web Service*

El servicio Web de MusicBrainz sólo permite realizar llamadas a través de REST con la operación GET.

Las respuesta de este servicio están en formato XML, anteriormente el servicio devolvía también RDF, pero, por considerarse complicado su uso, ha ido desapareciendo y recomiendan no utilizarlo salvo en proyectos que ya lo hagan y deban mantenerse con el mismo.

4.7.1. Llamadas

Las llamadas por lo tanto se van a realizar a través de peticiones REST de tipo GET. En este tipo de peticiones los datos que se le envían al servicio viajan de manera visible en la URL.

Como se dijo con anterioridad, MusicBrainz acepta, por cada tipo de entidad, dos consultas posibles, una por nombre en la que se muestran todas las coincidencias, y otra por el identificador que la entidad tiene en la base de datos de MusicBrainz, que por lo tanto muestra la información de una entidad concreta. Dado que Calais nos va a indicar qué entidades son Artistas, cuáles son Grupos y cuáles Álbumes, el tipo peticiones que vamos a poder realizar son las primeras, es decir a través del nombre. El caso es que esta primera consulta la única información que nos va a devolver a cerca del artista es poco más que su identificador en la base de datos de MusicBrainz. Por lo tanto, por cada entidad encontrada sobre la que se desee ampliar la información a través de MusicBrainz, se van a tener que realizar dos llamadas, una para conseguir el identificador y otra para el resto de información.

Las URLS de las llamadas se van a mostrar despiezadas en una tabla para comprender su contenido.

A continuación se muestra una tabla resumen del contenido de las tablas que van a ilustrar las URLs correspondientes a la primera llamada al servicio Web de MusicBrainz.

URL base	Tipo	Formato respuesta	Parámetro búsqueda
URL de acceso al servicio Web de MusicBrainz	Categoría de la entidad. **Opciones:** Artista, Álbum	Actualmente limitado a XML, pero hay que indicarlo	Parámetro por el que se desea realizar la búsqueda dentro de la base de datos de MusicBrainz. **Opciones:** Nombre, Título.

Tabla 2: Descripción campos URL primera llamada a MusicBrainz

La primera llamada por tanto va a necesitar indicar en los parámetros el nombre la entidad buscada, por ejemplo para un artista o grupo musical hay que indicar el nombre por el que buscar con el parámetro “name” y la URL además debe indicar que se desea buscar un artista añadiendo “/artist/”:

http://musicbrainz.org/ws/1/artist/?type=xml&name=Tori+Amos

URL base	Tipo	Formato respuesta	Parámetro búsqueda
http://musicbrainz.org/ws/1/	artist/	?type=xml	&name=Tori+Amos

Tabla 3: Análisis URL primera llamada a MusicBrainz para Artistas

Y para un álbum de música, e parámetro por el que realizar la búsqueda es el título del mismo “title” e indicar en la URL que se trata de un álbum “/release/”:

http://musicbrainz.org/ws/1/release/?type=xml&title=Help

URL base	Tipo	Formato respuesta	Parámetro búsqueda
http://musicbrainz.org/ws/1/	artist/	?type=xml	&name=Tori+Amos

Tabla 4: Análisis URL primera llamada a MusicBrainz para Álbumes

La segunda llamada, en cambio, indicará el identificador que tiene en la base de datos de MusicBrainz y las opciones de ampliación de información que se deseen añadir.

Esta es la tabla que describe el contenido de las tablas resumen que ilustran esta segunda llamada:

URL base	Tipo	MBID	Formato respuesta	Información de la respuesta
URL de acceso al servicio Web de MusicBrainz	Categoría de la entidad. **Opciones:** Artista, Álbum	Identificador de la entidad dentro de la base de datos de MusicBrainz	Actualmente limitado a XML, pero hay que indicarlo	Información que se desea obtener de la base de datos de MusicBrainz. Varía según la categoría de la entidad

Tabla 5: Descripción campos URL segunda llamada a MusicBrainz

En este caso para Artistas:

http://musicbrainz.org/ws/1/artist/c0b2500e-0cef-4130-869d-732b23ed9df5?type=xml&inc=url-rels+artist-rels+ratings

URL base	Tipo	MBID	Formato respuesta	Información de la respuesta
http://musicbrainz.org/ws/1/	artist/	c0b2500e-0cef-4130-869d-732b23ed9df5	?type=xml	&inc=url-rels+artist-rels+ratings

Tabla 6: Análisis URL segunda llamada a MusicBrainz para Artistas

Donde c0b2500e-0cef-4130-869d-732b23ed9df5 es el identificador que el artista tiene asociado en MusicBrainz. La información que se desea obtener se indica en el parámetro "inc", que en este caso se desea obtener las URLs relacionadas con el artista, "url-rels", las relaciones que tiene con otros artistas, "artista-rels", y la valoración que los usuarios de MusicBrainz han hecho del mismo, "rating".

A continuación se muestra una tabla resumen con la descripción de los valores incluidos en el parámetro inc para el caso concreto de las entidades tipo Artista:

Parámetro	Descripción
url-rels	Devuelve las URLS relacionadas con el artista
artist-rels	Devuelve las relaciones que tiene el artista con otros artistas
ratings	Valoración que los usuarios de MusicBrainz han hecho del artista. Valorado sobre 5 puntos

Tabla 7: Descripción valores del parámetro incluir de la URL de la segunda llamada a MusicBrainz para Artistas

Para Grupos Musicales aunque las llamadas se puedan hacer de misma manera que las de Artista, al hacerse a través de la misma URL (http://musicbrainz.org/ws/1/artist/), se ha considerado que sería interesante obtener información diferente para mostrar cómo realmente el sistema distingue entre los diferentes tipos de entidades, conservando así la semántica de que el servicio Web de Calais le ha dotado. Por tanto las llamadas se harán de esta manera:

http://musicbrainz.org/ws/1/artist/b10bbbfc-cf9e-42e0-be17-e2c3e1d2600d?type=xml&inc=url-rels+artist-rels+release-rels+ratings

URL base	Tipo	MBID	Formato respuesta	Información de la respuesta
http://musicbrainz.org/ws/1/	artist/	b10bbbfc-cf9e-42e0-be17-e2c3e1d2600d	?type=xml	&inc=url-rels+artist-rels+release-rels+ratings

Tabla 8: Análisis URL segunda llamada a MusicBrainz para Grupos Musicales

Donde b10bbbfc-cf9e-42e0-be17-e2c3e1d2600d es el identificador del grupo y las opciones que se van a incluir, a través del parámetro inc, en este caso son url-rels para mostrar las URLs relacionadas con el grupo, artista-rels para poder mostrar las relaciones de otros artistas con el grupo, release-rels para mostrar los discos que tiene en el mercado y ratings para ver la valoración de los usuarios de MusicBrainz sobre este grupo.

A continuación se muestra una tabla resumen con la descripción de los valores incluidos en el parámetro inc para el caso concreto de las entidades tipo Grupo Musical:

Parámetro	Descripción
url-rels	Devuelve las URLS relacionadas con el grupo musical
artist-rels	Devuelve las relaciones que tiene el grupo con otros artistas
release-rels	Devuelve los discos que el grupo ha sacado al mercado
ratings	Valoración que los usuarios de MusicBrainz han hecho del grupo. Valorado sobre 5 puntos

Tabla 9: Descripción valores del parámetro incluir de la URL de la segunda llamada a MusicBrainz para Grupos

Para Albums:

http://musicbrainz.org/ws/1/release/02232360-337e-4a3f-ad20-6cdd4c34288c?type=xml&inc= tracks+release-events+url-rels+ratings+artist

URL base	Tipo	MBID	Formato respuesta	Información de la respuesta
http://musicbrainz.org/ws/1/	release /	02232360-337e-4a3f-ad20-6cdd4c34288c	?type=xml	&inc= tracks+release-events+url-rels+ratings+artist

Tabla 10: Análisis URL segunda llamada a MusicBrainz para Álbumes

Donde 02232360-337e-4a3f-ad20-6cdd4c34288c es el identificador del álbum musical en MusicBrainz y el parámetro "inc" indica la información que se desea ampliar, como las URLs relacionadas con el álbum "url-rels", los artistas creadores, "artist", las versiones que se han sacado del mismo, "release-events", las pistas o canciones que incluye, "tracks", y la valoración que los usuarios de MusicBrainz han hecho del mismo, "rating".

A continuación se muestra una tabla resumen con la descripción de los valores incluidos en el parámetro inc para el caso concreto de las entidades tipo Álbum Musical:

Parámetro	Descripción
tracks	Devuelve las pistas o canciones de las que se compone el álbum musical
realese-events	Devuelve las versiones que han salido al mercado del álbum
url-rels	Devuelve las URLS relacionadas con el álbum musical
ratings	Valoración que los usuarios de MusicBrainz han hecho del álbum. Valorado sobre 5 puntos
artist	Artista o grupo autor del disco

Tabla 11: Descripción valores del parámetro incluir de la URL de la segunda llamada a MusicBrainz para Álbumes

4.7.2. Respuestas

La respuesta va a estar en XML, y como se ha visto en las llamadas, va a depender del tipo de llamada que se haga, en concreto:

Para búsquedas por nombre o título:

Devuelve un XML en el que hay una lista con los artistas/álbumes que comparten ese nombre/título. Dentro de esa lista cada artista está etiquetado por el tag <artist> dentro del cual se pueden observar dos atributo, uno de ellos será necesario para realizar la segunda llamada, o llamada por identificador, el id. Siempre se va a seleccionar el primer artista de la lista, ya que está ordenada por relevancia. Por ejemplo este fragmento de XML muestra el resultado de hacer la llamada:

http://musicbrainz.org/ws/1/artist/?type=xml&name=Tori+Amos

```
<metadata>
- <artist-list offset="0" count="65">
  - <artist type="Person" id="c0b2500e-0cef-4130-869d-732b23ed9df5" ext:score="100">
      <name>Tori Amos</name>
      <sort-name>Amos, Tori</sort-name>
      <life-span begin="1963-08-22"/>
    </artist>
  - <artist type="Group" id="7fb14de2-5787-42f2-aca0-389568e70570" ext:score="54">
      <name>赤い鳥</name>
      <sort-name>Akai tori</sort-name>
      <life-span end="1974" begin="1969"/>
    </artist>
```

Ilustración 16: XML de respuesta de la primera llamada a MusicBrainz para Artistas

En rojo aparecen resaltadas cada una de las respuestas o artistas que se corresponden con el nombre por el que se ha realizado la búsqueda. Como se ha dicho, se va a seleccionar siempre la primera, ya que van ordenados por relevancia.

En amarillo aparece resaltado el MBID que se va a seleccionar para realizar la segunda llamada.

En el caso de los álbumes ocurre lo mismo con la diferencia que las etiquetas del XML cambian de <artista-list> a <release-list> y de <artist> a <release> y de que dentro de esta última se muestra más información. En cualquier caso lo que se necesita extraer en esta respuesta para poder realizar la siguiente llamada es también el atributo id. Por ejemplo el fragmento de XML que se muestra a continuación es el resultado de hacer la llamada:

http://musicbrainz.org/ws/1/release/?type=xml&title=Help

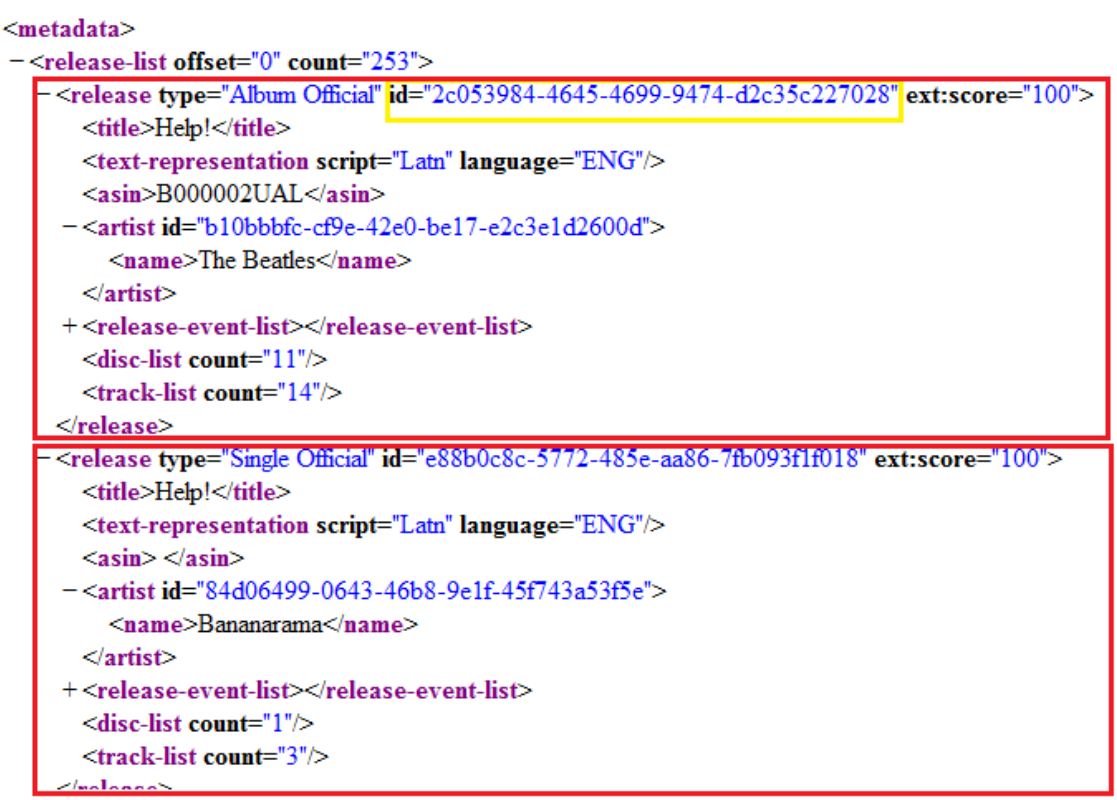


Ilustración 17: de respuesta de la primera llamada a MusicBrainz para Álbumes

En rojo aparecen resaltadas cada una de las respuestas o álbumes que se corresponden con el título por el que se ha realizado la búsqueda. Como se ha dicho, se va a seleccionar siempre la primera, ya que van ordenados por relevancia.

En amarillo aparece resaltado el MBID que se va a seleccionar para realizar la segunda llamada.

Para búsquedas por identificador de MusicBrainz o MBID:

El contenido de esta respuesta XML va a variar dependiendo de las opciones que aparezcan en la llamada bajo el parámetro inc. Estas opciones van variar dependiendo del tipo de entidad.

Para el caso de artista se va a requerir que se devuelva la siguiente información, que es la que sea considerado que puede tener más interés para el usuario:

- Las URLs que tengan relación con el artista, es decir la página de su club de fans, de su sitio oficial, la de Wikipedia, BBC Music, etc. De esta manera el usuario podrá ampliar la información del mismo si requiere conocer más de la que se le está mostrando con el sistema.

- Las relaciones con otros artistas, que nos va a devolver, si el artista tiene hijos artistas, una lista con sus nombres y MBID, además si el artista está o ha estado casado con algún artista también va a devolver una lista con sus nombres y MBIDs. Con esto se muestra al usuario un poco de información de la vida personal de artista y además se le permite enlazar con más información de estos otros artistas, ya que al disponer de sus MBID se muestra un enlace a la página de MusicBrainz dedicada a los mismos.

- La valoración que los usuarios de MusicBrainz han hecho del artista.

Se muestra a continuación un ejemplo en el que se puede observar cómo se muestra esta información. La respuesta en cuestión es el resultado de hacer la siguiente llamada:

http://musicbrainz.org/ws/1/artist/c0b2500e-0cef-4130-869d-732b23ed9df5?type=xml&inc=url-rels+artist-rels+ratings


```
<metadata>
 - <artist id="c0b2500e-0cef-4130-869d-732b23ed9df5" type="Person">
    <name>Tori Amos</name>
    <sort-name>Amos, Tori</sort-name>
    <life-span begin="1963-08-22"/>
    <rating votes-count="5">4.8</rating>
   - <relation-list target-type="Artist">
    - <relation type="Married" direction="backward" target="07538f3d-81d5-4c04-923e-4542b8ac9dbc" begin="1998-02-22" end="">
      - <artist id="07538f3d-81d5-4c04-923e-4542b8ac9dbc" type="Person">
          <name>Mark Hawley</name>
          <sort-name>Hawley, Mark</sort-name>
        </artist>
      </relation>
    </relation-list>
   - <relation-list target-type="Url">
      <relation type="BBCMusicPage" target="http://www.bbc.co.uk/music/artists/c0b2500e-0cef-4130-869d-732b23ed9df5" begin="" end=""/>
      <relation type="IMDb" target="http://www.imdb.com/name/nm0002169/" begin="" end=""/>
      <relation type="Wikipedia" target="http://en.wikipedia.org/wiki/Tori_Amos" begin="" end=""/>
      <relation type="Discogs" target="http://www.discogs.com/artist/Tori+Amos" begin="" end=""/>
      <relation type="Musicmoz" target="http://musicmoz.org/Bands_and_Artists/A/Amos,_Tori/" begin="" end=""/>
      <relation type="Fanpage" target="http://www.thedent.com/index.php" begin="" end=""/>
      <relation type="OfficialHomepage" target="http://www.toriamos.com/" begin="" end=""/>
      <relation type="Discography" target="http://www.yessaid.com/albums.html" begin="" end=""/>
      <relation type="Discography" target="http://www.hereinmyhead.com/" begin="" end=""/>
      <relation type="Myspace" target="http://www.myspace.com/toriamos" begin="" end=""/>
    </relation-list>
  </artist>
</metadata>
```


Ilustración 18: de respuesta de la segunda llamada a MusicBrainz para Artistas

Resaltado en color amarillo se observa en que parte del XML nos devuelve el valor “rating”, en rojo dónde y de qué manera lo hace para las relaciones con otros aristas y en verde las URLS relacionadas con el artista.

Si se trata de un grupo música, la información que se va a mostrar:

- Las URLs que tengan relación con el grupo, es decir la página de su club de fans, de su sitio oficial, la de Wikipedia, BBC Music, etc. Así el usuario podrá ampliar la información si necesitara obtener más de la que se le está mostrando con el sistema.

- Las relaciones con otros artistas, que nos va a devolver para este caso, los artistas que componen el grupo. De toda la información que se muestra al seleccionar esta opción se ha decidido agregar al sistema solo esta, ya que al ser un espacio reducido se ha considerado añadir sólo lo que pueda dar al usuario la información más importante. El resto de relaciones como las colaboraciones que se han hecho con otros artistas no se han considerado tan necesarias como para aparecer en un resumen.

- La valoración que los usuarios de MusicBrainz han hecho del grupo musical.

- Los discos que este grupo ha sacado al mercado, cada uno de ellos acompañado de un enlace a la página que MusicBrainz tenga dedicado a los mismos.

A continuación se muestra un ejemplo, donde la respuesta no aparece completa debido a su gran tamaño, pero aparecen cada una de estas opciones representadas:

http://musicbrainz.org/ws/1/artist/b10bbbfc-cf9e-42e0-be17-e2c3e1d2600d?type=xml&inc=url-rels+artist-rels+release-rels+ratings

Ilustración 19: de respuesta de la segunda llamada a MusicBrainz para Grupos

Se muestra resaltado en color amarillo dónde se encuentra dentro del XML el valor del "rating", en color rojo dónde están las relaciones del grupo con otros artistas (en ese caso los miembros del grupo), en color verde los discos que ha sacado el grupo y en color azul las URLS relacionadas con el mismo.

Para los álbumes musicales la información que se ha decidido más conveniente mostrar es:

- Las URLs que tengan relación con el álbum, es decir, enlace directo a Amazon para poder realizar su compra, la página de Wikipedia, la de BBC Music, etc. De esta manera el usuario podrá ampliar la información del mismo si requiere conocer más de la que se le está mostrando con el sistema o desea comprar el mismo.

- Las relaciones con otros artistas. De la información que muestra esta opción se va a seleccionar para mostrar sólo el grupo o el artista que haya sido responsable de su creación, es decir su autor.

- La lista de canciones que aparecen en el disco acompañadas de su duración y de un enlace a la página que MusicBrainz tenga dedicada a cada una de ellas.

- La valoración que los usuarios de MusicBrainz han hecho del álbum.

A continuación se muestra la respuesta procedente de la siguiente llamada, también de manera reducida al ser de gran tamaño:

http://musicbrainz.org/ws/1/release/02232360-337e-4a3f-ad20-6cdd4c34288c?type=xml&inc= tracks+release-events+url-rels+ratings+artist

Ilustración 20: de respuesta de la segunda llamada a MusicBrainz para Álbumes

Resaltado en color azul se muestra dónde aparece en el XML de respuesta la lista con los lanzamientos del disco, en color amarillo el “rating”, en rojo la lista de pistas o canciones que están incluidos en el disco y en color verde las URLS relacionadas con el disco.

4.8. *Librerías*

4.8.1. HelpBalloon

HelpBalloon se trata de una librería JavaScript gracias a la que se pueden añadir tooltips, es decir ventanas auxiliares con información, a cualquier elemento de un texto.

Basta con indicarle qué elemento del texto se desea comentar, qué contenido se desea mostrar en esta ventana externa y otros valores configurables como qué título debe tener la ventana emergente, si se desea que desaparezca de manera automática, tras cuánto tiempo y una serie de parámetros estéticos.

Esta librería permite que esta ventana (de ahora en adelante, tooltip) salga de diferentes tipos de elementos, como imágenes, íconos de información, links, etc. En este caso la venta debe emerger al pulsar sobre los links que aparezcan en el texto, que se tratará, como se puede intuir, de los elementos que ha reconocido Calais como pertenecientes al mundo de la música. Para ello lo único que se necesita es que cada uno de los elementos disponga de un identificador único para que HelpBalloon sepa sobre qué elemento desplegarse y qué información mostrar. Por esto es importante controlar las posibles repeticiones de los términos pertenecientes al texto en cuestión que han sido reconocidos por el servicio Web de Calais.

Además de este identificador, que en este caso será el elemento id de la etiqueta XML <A>, el tooltip de cada término debe llevar un título y el contenido a mostrar. El título va a coincidir con el nombre de la entidad reconocida y el contenido será la información adicional que haya devuelto MusicBrainz sobre la misma y que ha sido procesada y formateada previamente.

En concreto quedaría algo parecido a este ejemplo de los proporcionados por la Web que distribuye la librería:

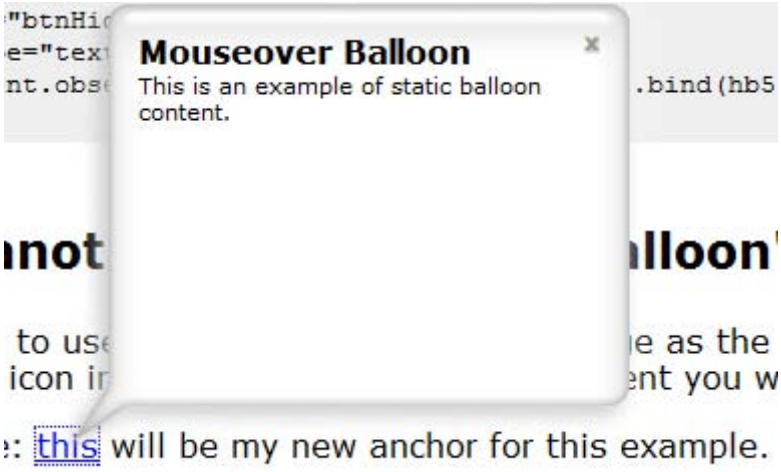


Ilustración 21: Ejemplo de tooltip usando la librería HelpBalloon

Para que aparezca de esta manera, las llamadas a la librería deben hacerse de esta manera por lo tanto:

var hb1 = new HelpBalloon({ title: artistName, content: content, icon: $(objectId) });

El contenido de los tooltips puede ser HTML, lo que va a facilitar la tarea a la hora de realizar una presentación amigable del mismo. Esto es importante ya que al tratarse de una ampliación de la información tampoco se requiere un texto denso, más bien texto resumido que pueda ser asimilado de un vistazo y con una interfaz amigable.

4.9. *Diseño de la implementación*

A continuación se va a detallar el diseño y desarrollo de los componentes del sistema a los que se hacía referencia en el apartado 5.4.

Componente Analizar Texto:

Este componente, como se ha comentado, se encarga de recoger el evento lanzado por el formulario y de transformarlo en una llamada real al servicio Web de Calais.

Para ello va a necesitar en primer lugar de la ayuda del objeto XMLHttpRequest para poder hacer la llamada POST necesaria para comunicar con este servicio. En concreto se creará un objeto XMLHttpRequest dependiendo del tipo de navegador en el que se esté ejecutando el código. Hay tres opciones, la primera que se trata de forma genérica, ya que el navegador que puede presentar problemas es el Internet Explorer, y las otras dos como alternativa en el caso de que aparezca un error durante la creación.

```
//This function creates the XmlHttpRequest object depending on the Web browser
function getXMLHTTPRequest(){
try{
  req = new XMLHttpRequest();
}catch(err1){
  try{
    req = new ActiveXObject("Msxm12.XMLHTTP");
  }catch(err2){
    try{
      req = new ActiveXObject("Microsoft.XMLHTTP");
    }catch(err3){
      req = false;
    }
  }
}

return req;
}
```

Ilustración 22: Código JavaScript para la creación de llamadas XMLHttpRequest

Una vez se disponga de este objeto hay que realizar la llamada al servicio Web. Esta llamada, debido a la ya mencionada limitación de la ejecución de scripts en dominios cruzados, se tiene que hacer a través del Proxy PHP.

Se establecen los parámetros que se van a enviar vía POST. El servicio Web necesita el número de licencia de uso del servicio Web, el contenido a analizar y el XML con los parámetros a indicar a Calais:

```
parameterString = 'licenseID='+licenseID+'&content='+content+'&paramsXML='+paramsXML
```

Ilustración 23: Código JavaScript para el establecimiento de los parámetros de Open Calais

Se utiliza el objeto XMLHttpRequest creado para realizar la llamada al servidor interno, a través del método open(), al que hay que indicarle el método, que en este caso es POST por motivos de capacidad, la URL, que es la del Proxy PHP, y un booleano que representa si la llamada es asíncrona (trae) o síncrona (false). En este caso se hará asíncrona para que no haya que recargar ni se interrumpa la visualización de la página. En la llamada al Proxy hay que indicar la URL del servicio Web dentro del parámetro action, el método POST en method y después una cadena con los parámetros de la URL a la que se va a redirigir en la forma nombre=valor, separados por &.

```
req.open('POST', 'http://localhost/calais/transport.php?action=
http://api.opencalais.com/enlighten/rest/&method=post&'+parameterString, true);
```

Ilustración 24: Código JavaScript para la llamada a Open Calais

Se realiza el envío de la petición a través de método send correspondiente:

```
req.send(parameterString);
```

Se espera a que llegue una respuesta. Si es de éxito, empezará el análisis:

```
req.onreadystatechange = function (aEvt) {
if (req.readyState == 4) {

  if(req.status == 200)  {
```

Ilustración 25: Código JavaScript para gestionar las llamadas asíncronas

Si se ha recibido el estado de éxito de http, es decir 200, se recoge la respuesta a través de la propiedad responseText del objeto XMLHttpRequest (se utilizará responseText ya que el formato devuelto es JSON) y se llama a uno de los métodos del componente Analizar JSON.

```
var textJson = req.responseText;
var result = evalJson(textJson);
```

Ilustración 26: Código JavaScript con la llamada a la función que evalúa la respuesta de Open Calais

La respuesta que éste devuelva se va a enviar al método underline() del componente Trasformar Texto:

```
underline(result,formName);
```

Ilustración 27: Código JavaScript con la llamada a la función que procesa la respuesta de Open Calais

- Ficheros JavaScript que lo implementan:
    - /javascrip/parseText.js
- Ficheros de los que depende:
    - /javascrip/XMLHttp.js
    - /javascrip/showText.js
    - transport.php

Componente Analizar JSON:

Este componente se va a encargar de analizar la respuesta JSON obtenida por el componente Analizar Texto como repuesta a la llamada al servicio Web Calais. Para ello, en primer lugar realizará una llamada a la función JavaScript, eval(), que evalúa este tipo de formato.

```
eval('json = ' + textJson);
```

Ilustración 28: Código JavaScript para evaluar la respuesta JSON

Después se hace una llamada a una función que resuelve las referencias intermedias, es decir las URIs se reemplazan en el objeto JSON por los objetos reales JSON a los que están apuntando.

```
function resolveReferences(flatdb) {
        for (var element in flatdb)
            for (var attribute in flatdb[element]) {
                var val = flatdb[element][attribute];
                if (typeof val == 'string')
                    if (flatdb[val] != null)
                        flatdb[element][attribute] = flatdb[val];
            }
}
```

Ilustración 29: Código JavaScript para resolver las referencias de la respuesta JSON

A continuación la respuesta pasa por otra función que crear un formato JSON más simple en el que las entidades son agrupadas en función de la categoría a la que pertenezcan:

```
function createHierarchy(flatdb) {
        var hdb = new Object();
        for (var element in flatdb) {
            var elementType = flatdb[element]._type;
            var elementGroup = flatdb[element]._typeGroup;
            if (elementGroup != null) {
                if (hdb[elementGroup] == null)
                    hdb[elementGroup] = new Object();
                if (elementType != null) {
                    if (hdb[elementGroup][elementType] == null)
                        hdb[elementGroup][elementType] = new Object();
                    hdb[elementGroup][elementType][element] = flatdb[element];
                } else
                    hdb[elementGroup][element] = flatdb[element];
            } else
                hdb[element] = flatdb[element];
        }
        return hdb;
}
```

Ilustración 30: Código JavaScript que devuelve agrupadas las categorías Artistas, Grupos y Álbumes

Una vez procesada la información JSON de esta manera, se procesa cada categoría de las encontradas por separado para almacenar las entidades agrupadas según estas categorías. Además se escoge la información que se necesita de entre toda la que ha devuelto Calais. En concreto para cada entidad se almacena, su nombre, la posición que tiene dentro del texto y la longitud de la misma. Estos dos últimos van a ser necesarios como coordenadas más tarde. Esto se realiza con el siguiente fragmento de código. Se muestra el de Grupos Musicales, pero se realiza la misma operación para Artistas y para Álbumes. Es decir se obtiene un array multidimensional por cada una de las categorías.

```
// Now, look for MusicGroups
obj = final_json.entities.MusicGroup;
if (obj != undefined)
{
   n = 0;       // Reset counter
      for(var musicgroup in obj)
   {
       //Look for repetitions of the same word
       for(var i=0;i<obj[musicgroup].instances.length;i++){
           groupNameList[n] = obj[musicgroup].name;
           groupPosList[n] = obj[musicgroup].instances[i].offset;
           groupLengthList[n] = obj[musicgroup].instances[i].length;
           n++;
       }
   }

   // Set multidimensional array for Music Group's information
   groupsList[0] = groupNameList;
   groupsList[1] = groupPosList;
   groupsList[2] = groupLengthList;
   groupNum = n;

}
```

Ilustración 31: Código JavaScript para procesar la respuesta JSON y extraer la información dependiendo de la categoría

- Ficheros JavaScript que lo implementan:
    - /javascrip/parse_JSON.js
- Ficheros de los que depende:
    - No depende de ninguno

Componente Transformar Texto:

Este componente se encarga de combinar el texto que ha introducido el usuario con la información que ha aportado el servicio Web de Calais y que ha sido procesada por el componente AnalizarTexto. Va a reemplazar cada una de las entidades reconocidas por un enlace HTML que realice una llamada al método que llama al servicio Web de MusicBrainz para ampliar información.

Recibe el array tridimensional de AnalizarTexto, que, como se ha dicho, está formado por cuatro arrays, uno por cada tipo de entidad y uno más que tiene la longitud de los mismos. Se procesa cada array de entidades por separado para poder realizar las llamadas convenientes según su categoría. Siempre van a venir en el mismo orden y, en el caso de que falte alguno, como se conocen las longitudes, no se considera el mismo y se pasa al siguiente.

```
counter = 0;
//Once for artist, once for group and once for album
for (var i=0; i < 3; i++){
    var elements = result[numClasses-1][i];
    if(result[numClasses-1][i]>0)
    {
        //Once for every entity
        for(var j=0; j < elements; j++){

            var oldWord = result[counter][0][j];
            var paramName = oldWord.split(' ');
            var name = "";
            var typeStr;
            for (var k=0;k<paramName.length;k++){
                name = name+paramName[k];
            }
            var repit = nameElement(result,counter,j,oldWord,numArray);
            name = name+repit;
            switch (i){
                case 0:
                    var info = oldWord+'$'+name+'$artist';
                    //info = escape(info);
                    newWord = '<A id="'+name+'" HREF="javascript:goMusicBrainzAsync(\''+ info +'\')">'+ oldWord+'</A>';
                    allArtists[numArtists] = info;
                    numArtists++;
                break;
```

Ilustración 32: Código JavaScript para crear los links de las entidades reconocidas dentro del texto

Se puede observar en el código que hay un switch para elegir qué llamada hacer dependiendo de la categoría. Para los casos en los que no haya representantes de alguna categoría, al llevar un contador a parte (counter) que sólo se incrementa si el tamaño del array (se obtiene a través del último array que lleva los tamaños de los tres) es mayor que cero, por lo que no hay peligro de desbordamiento y se mantiene la coherencia del switch para cada caso. Sólo aparece el switch para el caso 0, pero sería lo mismo para los otros dos, variando la formación de la nueva palabra.

Este switch reemplaza la entidad tal y como aparece en el texto del usuario, por un enlace HTML al método que realiza la ampliación de la información. Esto conlleva el inconveniente de que el sistema es capaz de situar las palabras en el texto a través de unas coordenadas, al cambiar las entidades por enlaces estas coordenadas cambian. Esto se hace a través de la siguiente función, que sólo actualiza las coordenadas de las entidades que se vean afectadas por estas colocadas en el texto detrás de la que haya provocado el cambio:

```
function moveForward(entitiesArray,type,pos,newPos,init,num){

    //var counter = type;
    var n = entitiesArray.length-1;
    var tam = 0;
    for (var i=type; i<entitiesArray.length-1;i++){
        var from = 0;
        if(i==type)
            from=pos+1;
        else
            from = 0;
        tam = num[i];
        for (var j=from; j<tam; j++){
            //Changes positions only if it goes after the changed word
            if(init < entitiesArray[i][1][j]){
                entitiesArray[i][1][j] = entitiesArray[i][1][j]+newPos;
            }
        }
    }
    return entitiesArray;
}
```

Ilustración 33: Código JavaScript para avanzar las coordenadas de la entidad dentro del texto

Al final va a devolver un array multidimensional, que va a estar compuesto por cada uno de los arrays y como elemento final un array con el tamaño de cada uno de ellos. Este último array va a resultar útil a la hora de procesar estos resultados ya que podría haber confusiones si una de las categorías no tiene representantes en el texto que se analice.

```
var elements = new Array();
elements[0] = artistNum;
elements[1] = groupNum;
elements[2] = albumNum;
var num = 0;
if(artistNum > 0){
  result[num] = artistsList;
  num++;
  }
  if(groupNum > 0){
      result[num] = groupsList
      num++;
  }
  if(albumNum > 0){
      result[num] = albumsList
      num++
  }

result[num] = elements
return result;
```

Ilustración 34: Código JavaScript para almacenar los tamaños de los arrays de categorías

Otra función realiza una búsqueda recorriendo todos los arrays del array multidimensional para comprobar si alguna de las entidades está repetida, aunque sea en diferente categoría. Una vez conocido el número de repeticiones se anota en el identificador que va a llevar el enlace HTML correspondiente. Esto es necesario ya que la librería utilizada para mostrar los tooltips necesita que cada una de las entidades tenga un identificador único, por lo que se ha decidido que sea: nombre_entidad+num_repeticiones, donde num_repeticiones es el número de coincidencias con el nombre de la entidad desde la posición que éste tiene en el array multidimensional en la que se encuentra hasta el final del mismo.

```
function nameElement(entitiesArray,type,pos,name,num){
    var repit = 0;
    //var counter = type;
    var tam = 0;
    for (var i=type; i<entitiesArray.length-1;i++){

        var from = 0;
        if(i==type)
            from=pos+1;
        else
            from = 0;
        tam = num[i];
        for (var j=from; j<tam; j++){
            if (name.indexOf(entitiesArray[i][0][j])!=-1){
                repit++;
            }
        }
    }
    return repit;
}
```

Ilustración 35: Código JavaScript para comprobar las repeticiones de una entidad dentro del texto

Una vez se conoce cuál es la sustitución de la entidad dentro del texto, se hace una primera llamada al componente FormatearArtista, FormatearGrupo o FormatearAlbum, dependiendo de la categoría de la entidad. Estas llamadas son necesarias para que se carguen todos los tooltips, ya que si no habría que esperar un tiempo considerable desde que el usuario pulsara sobre el enlace de la entidad a consultar hasta que apareciera la información y podría dar lugar a confusiones al usuario.

Se pude observar como entre llamadas hay una llamada a la función sleep, con valor de un segundo. Esto es una imposición del servicio Web MusicBrainz que no permite realizar más de una llamada por segundo, devolviendo un error si se incumple.

```
for (var i = 0; i<allArtists.length;i++){
    var wait = true;
    wait = goMusicBrainz(allArtists[i]);
    sleep(1000);
}
for (var i = 0; i<allAlbums.length;i++){
    var wait = true;
    wait = goMusicBrainzAlbum(allAlbums[i]);
    sleep(1000);
}
```

Ilustración 36: Código JavaScript que hace las llamadas a la función que hace las llamadas al servicio Web de MusicBrainz

- Ficheros JavaScript que lo implementan:
    - /javascrip/manage_text.js
- Ficheros de los que depende:
    - /javascrip/syncCalls.js
    - /javascrip/asyncCalls.js
    - /javascrip/utils.js

Componente Formatear Artista:

Este componente se encarga de realizar la llamada al servicio Web de MusicBrainz, recoger la información adecuada, transformarla y mostrarla.

De este componente hay dos variantes, una de ellas que realiza las llamadas con el objeto XMLHttpRequest en su variante síncrona y la otra la asíncrona. El uso de una o de otra va a depender del momento de su invocación.

La síncrona va a ser llamada desde el componente TransformarTexto, ya que este componente realiza una gran cantidad de llamadas consecutivas y la variante asíncrona se hace un lío con los estados que devuelve http y no espera a recibir el estado de éxito. Con la variante síncrona no ocurre esto ya que no se realiza una llamada hasta que no se devuelve la información de la anterior.

```
req.open('GET', 'http://localhost/calais/transport.php?action='+urlBase+'&method=get&'+urlSearch, false);
req.setRequestHeader('Content-Type', 'text/xml');
req.send(null);
```

Ilustración 37:Código JavaScript para hacer las llamadas síncronas a MusicBrainz

La llamada se hace a través del Proxy PHP a la URL de MusicBrainz, indicada en el parámetro urlBase, el método, es el definido por el servicio Web, es decir, get. Además se le añaden los parámetros que sean necesarios, en este caso el nombre de la entidad. El booleano que describe el tipo de llamada, se pone a false al tratarse de una llamada síncrona.

```
var checkOk=req.responseText;
if(checkOk.indexOf("The MusicBrainz web server is currently busy")!=-1){
    sleep(1000);
    goMusicBrainz(repit);

}else{
    artistId = req.responseXML.documentElement.getElementsByTagName("artist")[0].attributes.getNamedItem("id").nodeValue;
    urlGetInfo = artistId;
    sleep(1000);
    //Shows different info depending on the kind of object -- artist or group
    if(params[2]=='artist'){
        wait = getUrlsArtist(urlGetInfo,urlBase,name,params[1]);
        return wait;
    }else{
        if(params[2]=='group'){
            getUrlsGroup(urlGetInfo,urlBase,name,params[1]);
            return wait;
        }else{
            alert("Se ha producido un error de procesamiento. Por favor vuelva a intentarlo");
            return wait;
        }

    }
}

}
```

Ilustración 38: Código JavaScript para hacer la segunda llamada a MusicBrainz dependiendo de la categoría

El contenido se recoge coge con la propiedad responseText primero por si ha devuelto algún error de procesamiento. Si no, lo vuelve a recoger, esta vez con responseXML.

En esta llamada entra por primera vez el uso de DOM. Se recoge del XML de respuesta el identificador del artista o grupo. Después comprueba si se trata de un artista o de un grupo y realiza la segunda llamada al servicio Web de MusicBrainz para conseguir la información adicional de la entidad.

La asíncrona se utiliza cuando, al haber ocurrido un error inesperado durante la ejecución del componente Transformar Texto, no se tiene información de una entidad y es el usuario el que al pulsar sobre el enlace de la entidad realiza la llamada. Lo único que cambia en ella es la manera de realizar la llamada, aquí se utiliza el valor true en el booleano y que se mantiene esperando a que cambie el estado http:

```
req=getXMLHTTPRequest();
req.open('GET','http://localhost/calais/transport.php?action='+urlBase+urlGetInfo+'&method=get&type=xml&inc=url-rels+artist-rels+ratings', true);
req.setRequestHeader('Content-Type', 'text/xml');
req.onreadystatechange = function (aEvt) {
if (req.readyState == 4) {

    if(req.status == 200)  {
```

Ilustración 39: Código JavaScript para gestionar la respuesta de las llamadas asíncronas

La segunda llamada es la que se hace indicando como argumento el MBID en vez del nombre de la entidad.

```
req.open('GET','http://localhost/calais/transport.php?action='+urlBase+urlGetInfo+
'&method=get&type=xml&inc=url-rels+artist-rels+ratings', false);
```

Ilustración 40: Código JavaScript para hacer la segunda llamada de la categoría Artista

La respuesta será diferente según el tipo de entidad y por lo tanto se procesará también de manera diferente. Este componente concreto recoge la siguiente información del artista por medio DOM:

```
var artistRels = xmlDoc.getElementsByTagName("relation-list")[i].childNodes;
for(var j=0;j<artistRels.length;j++){
    var relationType = xmlDoc.getElementsByTagName("relation-list")[i].childNodes[j].attributes.getNamedItem("type").nodeValue;
    var relationContent = xmlDoc.getElementsByTagName("relation-list")[i].childNodes[j].childNodes[0].childNodes[0];
    switch (relationType){
```

Ilustración 41: Código JavaScript para extraer la información de un Artista

Lo primero es recoger cada una de las relaciones con otros artistas, seleccionando el tipo y el contenido del mismo.

Dependiendo del tipo se va a recoger la información o no. En este caso serán hijos, matrimonios y bandas de las que ha sido miembro.

```
switch (relationType){
    case "Parent":
        childLinks[childNumber] = xmlDoc.getElementsByTagName("relation-list")[i].childNodes[j].childNodes[0].attributes.getNamedItem("id").nodeValue;
        childNames[childNumber] = relationContent.childNodes[0].nodeValue;
        childNumber = childNumber + 1;
    break;
    case "Married":
        marriageLinks[marriageNumber] = xmlDoc.getElementsByTagName("relation-list")[i].childNodes[j].childNodes[0].attributes.getNamedItem("id").nodeValue
        marriageNames[marriageNumber] = relationContent.childNodes[0].nodeValue;
        marriageNumber = marriageNumber + 1;
    break;
    case "MemberOfBand":
        groupLinks[groupNumber] = xmlDoc.getElementsByTagName("relation-list")[i].childNodes[j].childNodes[0].attributes.getNamedItem("id").nodeValue;
        groupNames[groupNumber] = relationContent.childNodes[0].nodeValue;
        groupNumber = groupNumber + 1;
    break;
}
```

Ilustración 42: Código JavaScript para formatear la información de un artista dependiendo del contenido de la misma

Una vez haya sido recogida esta información se prepara para ser mostrada formateándola como si de texto HTML se tratara.

Si el tipo de la etiqueta que acompaña al artista es URL, es que se trata de las URLS relacionadas con el mismo. Se recogen con DOM todas las que haya.

```
if(targetType == "Url"){
    //Sets the additional info about the artist, in other words, the relating Urls
    var urlRels = xmlDoc.getElementsByTagName("relation-list")[i].childNodes;
    content += '<table border=0><tr>';
```

Ilustración 43: Código JavaScript para formatear las URLS de la entidad

Cada una de las diferentes URLS llevan un identificador dependiendo de la Web que las sirva y es sobre este identificador sobre el que se va a decidir si se incluye dentro del tooltip o no. Esta diferenciación se va a hacer a través de un switch. A continuación se muestra parte del código:

```
for(var j=0;j<urlRels.length;j++){
    var artistUrl = xmlDoc.getElementsByTagName("relation-list")[i].childNodes[j].attributes.getNamedItem("target").nodeValue;
    var urlName = xmlDoc.getElementsByTagName("relation-list")[i].childNodes[j].attributes.getNamedItem("type").nodeValue;
        switch (urlName){
            case "Wikipedia":
                content += '<td><A HREF='+artistUrl+' target="_blank"><img src="./images/wikipedia.bmp" /></A> </td>'
            break;
            case "Myspace":
                content += '<td><A HREF='+artistUrl+' target="_blank"><img src="./images/myspace.bmp" /></A> </td>'
            break;
```

Ilustración 44: Código JavaScript para añadir a las URLS de la entidad una imagen para identificarlos

Una vez recogida toda esta información se crea el tooltip para la entidad diciéndole cuál va a ser el título, el contenido y el identificador del elemento sobre el que se va a abrir:

```
var hb1 = new HelpBalloon({ title: artistName, content: content, icon: $(objectId) });
```

Ilustración 45: Código JavaScript para crear el tooltip

- Ficheros JavaScript que lo implementan:
    - /javascrip/sync_calls.js
    - /javascrip/async_calls.js
- Ficheros de los que depende:
    - /javascrip/XMLHttp.js
    - /javascrip/utils.js

Componente Formatear Grupo:

Este componente se encarga de realizar la llamada al servicio Web de MusicBrainz para preguntar sobre un Grupo Musical, recoger la información adecuada, transformarla y mostrarla.

De este componente también hay dos variantes, la que realiza las llamadas con el objeto XMLHttpRequest en su variante síncrona y la que lo hace con la asíncrona. El uso de una o de otra también va a depender del momento de su invocación.

La síncrona va a ser llamada desde el componente TransformarTexto, ya que este componente realiza una gran cantidad de llamadas consecutivas y la variante asíncrona se hace un lío con los estados que devuelve http y no espera a recibir el estado de éxito. Con la variante síncrona no ocurre esto ya que no se realiza una llamada hasta que no se devuelve la información de la anterior.

Esta llamada es común con la del componente Formatear Artista, ya que MusicBrainz no hace distinción entre ellos, será más adelante en el componente donde se diferencien.

```
req.open('GET', 'http://localhost/calais/transport.php?action='+urlBase+'&method=get&'+urlSearch, false);
req.setRequestHeader('Content-Type', 'text/xml');
req.send(null);
```

Ilustración 46: Código JavaScript para realizar la llamada de a MusicBrainz para Grupos

Aquí se puede observar el punto en el que se realiza la ramificación entre ambos, que coincide con el momento en el que se va a efectuar la segunda llamada al servicio MusicBrainz para conseguir la información extra de la entidad.

```
var checkOk=req.responseText;
if(checkOk.indexOf("The MusicBrainz web server is currently busy")!=-1){
    sleep(1000);
    goMusicBrainz(repit);

}else{
    artistId = req.responseXML.documentElement.getElementsByTagName("artist")[0].attributes.getNamedItem("id").nodeValue;
    urlGetInfo = artistId;
    sleep(1000);
    //Shows different info depending on the kind of object -- artist or group
    if(params[2]=='artist'){
        wait = getUrlsArtist(urlGetInfo,urlBase,name,params[1]);
        return wait;
    }else{
        if(params[2]=='group'){
            getUrlsGroup(urlGetInfo,urlBase,name,params[1]);
            return wait;
        }else{
            alert("Se ha producido un error de procesamiento. Por favor vuelva a intentarlo");
            return wait;
        }

    }
}

}
```

Ilustración 47: Código JavaScript para diferenciar entre las llamadas de Artista y Grupo

La segunda llamada es la que se hace indicando como argumento el MBID en vez del nombre de la entidad.

```
req.open('GET','http://localhost/calais/transport.php?action='+urlBase+urlGetInfo+
'&method=get&type=xml&inc=url-rels+artist-rels+release-rels+ratings', false);
```

Ilustración 48: Código JavaScript para realizar la segunda llamada a MusicBrainz demandando la información necesaria para Grupos

La diferencia con la hecha en el componente Formatear Artista en las opciones que se desean mostrar, esto se va a notar en las opciones que se van a indicar en el argumento inc.

A continuación se muestran las diferencias con el componente de Artistas a la hora de recoger la información procedente del servicio MusicBrainz. Lo primero es recoger la información referente a la de otros artistas relacionados con el grupo, que para esta entidad va a ser los miembros del mismo:

```
var targetType = xmlDoc.getElementsByTagName("relation-list")[i].attributes.getNamedItem("target-type").nodeValue;
if(targetType == "Artist"){
    //Sets the artists info
    var artistRels = xmlDoc.getElementsByTagName("relation-list")[i].childNodes;
    for(var j=0;j<artistRels.length;j++){
        var relationType = xmlDoc.getElementsByTagName("relation-list")[i].childNodes[j].attributes.getNamedItem("type").nodeValue;
        var relationContent = xmlDoc.getElementsByTagName("relation-list")[i].childNodes[j].childNodes[0].childNodes[0];
        switch (relationType){
            case "MemberOfBand":
                groupNames[groupNumber] = relationContent.childNodes[0].nodeValue;
                groupNumber = groupNumber + 1;
            break;
        }
    }
```

Ilustración 49: Código JavaScript para extraer la información para los Grupos

De esta entidad también va a ser recogida la información relativa a los discos que ha lanzado al mercado:

```
if(targetType == "Release"){
    var releaseNames = new Array();
    var releaseLinks = new Array();
    var releaseTypes = new Array();
    var artistReleases = xmlDoc.getElementsByTagName("relation-list")[i].childNodes;
    for(var j=0;j<artistReleases.length;j++){
        releaseTypes[j] = xmlDoc.getElementsByTagName("relation-list")[i].childNodes[j].childNodes[0].attributes.getNamedItem("type").nodeValue;
        releaseNames[j] = xmlDoc.getElementsByTagName("relation-list")[i].childNodes[j].childNodes[0].childNodes[0].childNodes[0].nodeValue;;
        releaseLinks[j] = xmlDoc.getElementsByTagName("relation-list")[i].childNodes[j].childNodes[0].attributes.getNamedItem("id").nodeValue;
    }
```

Ilustración 50: Código JavaScript para procesar la información de los discos del grupo

Como para el componente Formatear Artista se seleccionan las URLs relacionadas con el Grupo, procedentes de determinados sitios Web:

```
if(targetType == "Url"){
    //Sets the additional info about the artist, in other words, the relating Urls
    var urlRels = xmlDoc.getElementsByTagName("relation-list")[i].childNodes;
    content += '<table border=0><tr>';
    for(var j=0;j<urlRels.length;j++){
        var artistUrl = xmlDoc.getElementsByTagName("relation-list")[i].childNodes[j].attributes.getNamedItem("target").nodeValue;
        var urlName = xmlDoc.getElementsByTagName("relation-list")[i].childNodes[j].attributes.getNamedItem("type").nodeValue;
        switch (urlName){
            case "Wikipedia":
                content += '<td><A HREF='+artistUrl+' target="_blank"><img src="./images/wikipedia.bmp" /></A> </td>'
            break;
            case "Myspace":
                content += '<td><A HREF='+artistUrl+' target="_blank"><img src="./images/myspace.bmp" /></A> </td>'
            break;
```

Ilustración 51: Código JavaScript para formatear las URLS de Grupos

Se prepara el tooltip con la información pertinente, título, contenido e identificador de la entidad sobre la que se va a desplegar:

```
var hb1 = new HelpBalloon({ title: groupName, content: content, icon: $(objectId) });
```

Ilustración 52: Código JavaScript para crear el tooltip con la información de los Grupos

- Ficheros JavaScript que lo implementan:
    - /javascrip/sync_calls.js
    - /javascrip/async_calls.js
- Ficheros de los que depende:
    - /javascrip/XMLHttp.js
    - /javascrip/utils.js

Componente Formatear Álbum:

Este componente funciona como sus análogos para las entidades Artista y Grupo. Realiza también dos tipos de llamadas, las síncronas y las asíncronas, en las mismas situaciones. Va a variar la URL a la que se realizan las llamadas, como se puede ver en la variable que la define:

```
var urlBase = 'http://musicbrainz.org/ws/1/release/';
```

Ilustración 53: Código JavaScript para establecer la URL de llamada a MusicBrainz para Álbumes

La segunda llamada, o llamada a través del MBID, además de hacerse a una URL diferente va a llevar opciones de información distintas a la de sus análogos.

```
req.open('GET','http://localhost/calais/transport.php?action='+urlBase+urlGetInfo+
'&method=get&type=xml&inc=tracks+release-events+url-rels+ratings+artist', false);
```

Ilustración 54: Código JavaScript para realizar la llamada a MusicBrainz con la información para Álbumes

Se van a recoger las pistas que aparecen en el disco en cuestión, a través de DOM una vez más:

```
for(var i=0;i<albumTracks.length;i++){
    var track = xmlDoc.getElementsByTagName("track-list")[0].childNodes[i];
    var trackName = track.childNodes[0].childNodes[0].nodeValue;
    var duration = track.childNodes[1].childNodes[0].nodeValue;
    var trackNumber = i + 1;
    var trackMinutes = getMinutes(duration);
    var trackId = track.attributes.getNamedItem("id").nodeValue;
    var mbLink = "http://musicbrainz.org/track/"+trackId+".html";
```

Ilustración 55: Código JavaScript para formatear la información de las canciones que forman el Álbum

Con las URLs se va a proceder como ha hecho en sus análogos:

```
if (hasUrl.length > 0){
    var urlRels = xmlDoc.getElementsByTagName("relation-list")[0].childNodes;
    content += '<table border=0><tr>';
    for(var i=0;i<urlRels.length;i++){
        var artistUrl = xmlDoc.getElementsByTagName("relation-list")[0].childNodes[i].attributes.getNamedItem("target").nodeValue;
        var urlName = xmlDoc.getElementsByTagName("relation-list")[0].childNodes[i].attributes.getNamedItem("type").nodeValue;
        switch (urlName){
            case "Wikipedia":
                content += '<td><A HREF='+artistUrl+' target="_blank"><img src="./images/wikipedia.bmp" /></A> </td>'
            break;
            case "Myspace":
                content += '<td><A HREF='+artistUrl+' target="_blank"><img src="./images/myspace.bmp" /></A> </td>'
            break;
```

Ilustración 56:Código JavaScript para formatear las URLS del Álbum

Una vez recogida la información a través de DOM y formateada con HTML se crea el tooltip, con el título, el contenido y el identificador único de la entidad:

```
var hb1 = new HelpBalloon({ title: albumName, content: content, icon: $(objectId) });
```

Ilustración 57: Código JavaScript para crear el tooltip con la información del Álbum

- Ficheros JavaScript que lo implementan:
    - /javascrip/sync_calls.js
    - /javascrip/async_calls.js
- Ficheros de los que depende:
    - /javascrip/XMLHttp.js
    - /javascrip/utils.js

Componente HelpBalloon:

Componente externo al sistema para la creación de los tooltips.

Ficheros que lo implementan:

- Paquete /HelpBalloon/*

4.10. *Entorno de ejecución*

Para el entorno de ejecución este es el software necesario:

- Servidor Apache 2.2 (http://httpd.apache.org/download.cgi)

- PHP versión 5 (http://php.net/downloads.php)

Para el entorno de pruebas, por comodidad, se decidió utilizar una solución paquetizada, llamada WamServer en su versión 2.0i (07/11/09) que incluye:

- Apache 2.2.11
- MySQL 5.1.36
- PHP 5.3.0

(http://www.wampserver.com/en/download.php)

Para poder ejecutar el código JavaScript sólo es necesario un navegador Web. Para este sistema se va a asegurar su funcionamiento en los siguientes navegadores:

- Internet Explorer
- Mozilla Firefox

Si se desea obtener más información a cerca de la puesta en marcha del entorno de ejecución se puede consultar el **Anexo I: Manual de usuario.**

4.11. *Pruebas realizadas*

A continuación se indican una serie de pruebas de aceptación del sistema que se han realizado, mostrando el texto de partida y cómo quedaría transformado.

Las primeras pruebas se corresponden con casos ideales, para, conforme vayan avanzando, mostrar casos más problemáticos y así comprobar que se ha construido un sistema robusto.

La tabla que se muestra a continuación muestra un resumen de las pruebas realizadas con los resultados obtenidos:

	Identificador	**Objetivos**	**Resultados**
Prueba 01	Prueba de uso completo	Comprobar: - El reconocimiento de entidades de todas las categorías. - El reconocimiento de las entidades repetidas. - La información mostrada en el tooltip. - Los tiempos.	- Reconoce OK - Repeticiones OK - Información coherente con la disponible - Dependencia del tiempo de transporte
Prueba 02	Prueba de uso limitado a una sola categoría	Comprobar: - La capacidad para reconocer entidades de una sola categoría. - Los tiempos.	- Reconoce OK - Dependencia del tiempo de transporte.
Prueba 03	Prueba de uso limitado a dos categorías	Comprobar: - La capacidad para reconocer entidades de dos categorías. - La coherencia semántica entre categorías. - Los tiempos	- Reconoce OK - Mantiene la semántica de Calais - Dependencia del tiempo de transporte.
Prueba 04	Prueba de ambigüedad – contexto 1	Comprobar: - El reconocimiento de acrónimos. - La desambiguación.	- Acrónimos registrados musicalmente OK - Depende de Calais
Prueba 05	Prueba de ambigüedad – contexto 2	Comprobar: - La desambiguación	- Calais depende del

		de la Prueba 04 en otro contexto.	contexto.
Prueba 06	Prueba de texto procedente de otra fuente	Comprobar: - Reconocimiento de texto no procedente de la Wikipedia.	- Reconoce OK
Prueba 07	Prueba de nombres incompletos	Comprobar: - Reconocimiento de nombres incompletos.	- Reconoce OK si aporta información el contexto.
Prueba 08	Prueba de nombres incompletos no reconocidos	Comprobar: -Reconocimiento de nombres incompletos en otro texto	- El reconocimiento re nombres incompletos depende de la información extra del texto.

Tabla 12: Resumen de las pruebas realizadas sobre el sistema

Prueba de uso completo: Texto perteneciente al contexto musical con al menos un representante de cada categoría, los cuales se tratan entidades famosas dentro del panorama musical. El texto ha sido extraído de Wikipedia.

Objetivos:

- Comprobar que es capaz de reconocer entidades y diferenciarlas dependiendo de la categoría a la que pertenecen.
- Comprobar que si hay más de una repetición de una misma entidad, muestra la misma información para todas ellas.
- Comprobar que la información mostrada en el tooltip es coherente con la disponible en MusicBrainz sobre la entidad. Es decir que si no se dispone de determinada información no se haga referencia a ella, por ejemplo, mostrando tablas o lisas vacías.
- Comprobar el tiempo empleado por el sistema en completar la solución.

Texto:

The Beatles were an English rock band, formed in Liverpool in 1960, and one of the most commercially successful and critically acclaimed acts in the history of popular music.[1] From 1962 the group consisted of John Lennon (rhythm guitar, vocals), Paul McCartney (bass guitar,

vocals), George Harrison (lead guitar, vocals) and Ringo Starr (drums, vocals).

Beatlemania and touring years (1963–1966)

UK popularity, Please Please Me and With The Beatles

McCartney, Harrison, Swedish pop singer Lill-Babs and Lennon on the set of the Swedish television show Drop-In, 30 October 1963

In the wake of the moderate success of "Love Me Do", "Please Please Me" met with a more emphatic reception, reaching number two in the UK singles chart after its January 1963 release. Martin originally intended to record the band's debut LP live at The Cavern Club. Finding it had "the acoustic ambience of an oil tank",[45] he elected to create a "live" album in one session at Abbey Road Studios. Ten songs were recorded for Please Please Me, accompanied on the album by the four tracks already released on the two singles.

Resultado:

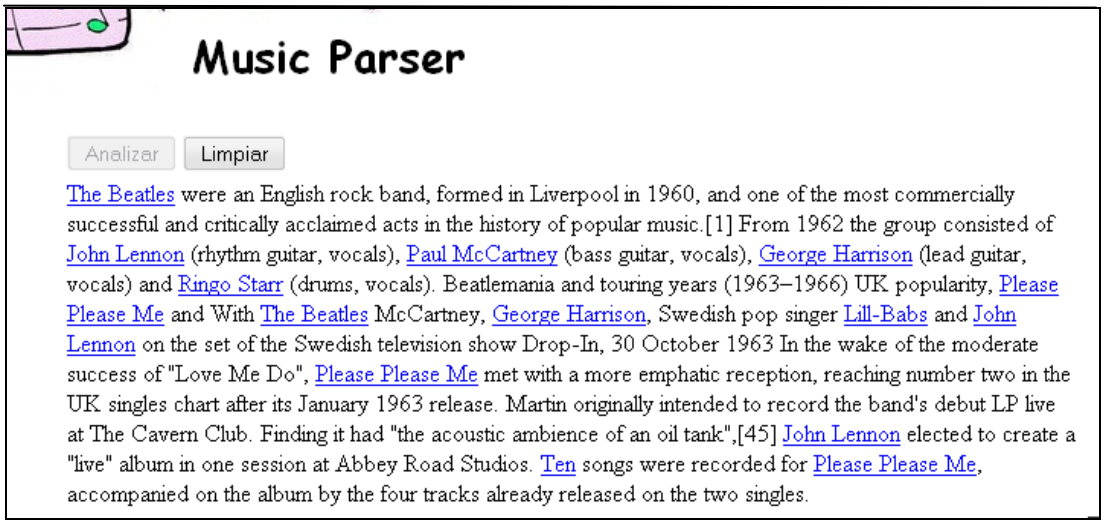


Ilustración 58: Resultado Prueba 01

Se comprueba el texto reconocido y se observa que OpenCalais no ha sido capaz de reconocer “Love Me Do”, en la línea 7, al no tratarse de un álbum, si no de un single, tampoco ha sido capaz de reconocer a “Paul McCartney”, línea 5, únicamente por su apellido. Otro fallo que ha tenido es el de reconocer “Ten”, línea 10, como grupo de música cuando realmente en este contexto no se refiere a él, si no a ten como número. El resto de entidades ha sido capaz de reconocerlas de manera adecuada y se ha comprobado que se han clasificado correctamente según los tipos Artista, Grupo Musical y Álbum Musical.

Se comprueba ahora el contenido de las entidades reconocidas.

Se selecciona una entidad de tipo Artista y se comprueba que dispone de toda la información posible: hijos artistas, matrimonios con artistas, grupos de los que ha formado parte, enlaces relacionados con el artista y valoración que los usuarios de MusicBrainz hacen del mismo.

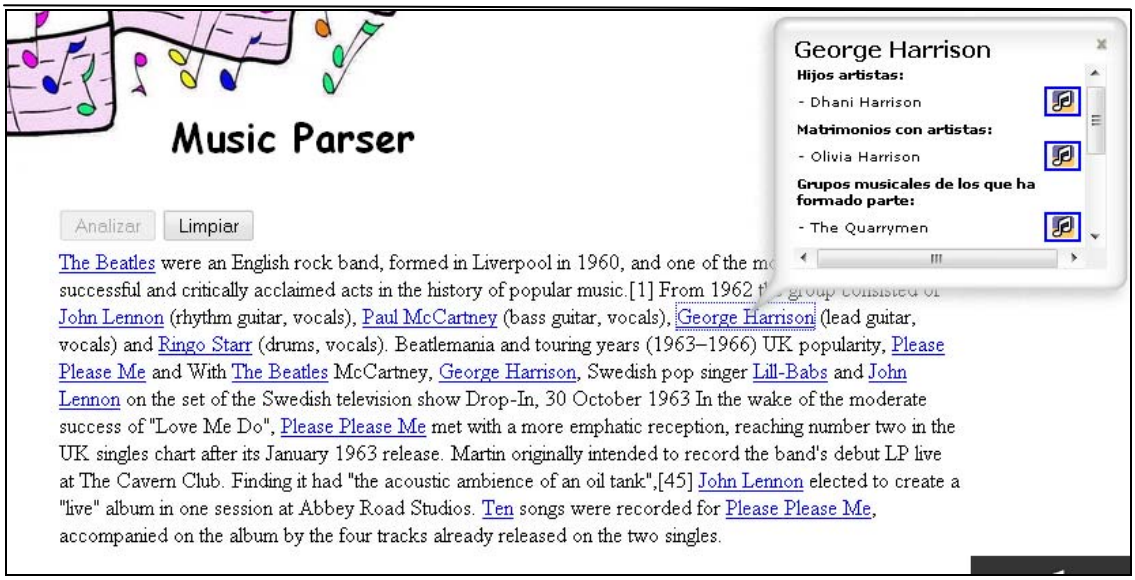


Ilustración 59: Resultado Prueba 01. Contenido del tooltip para Artista

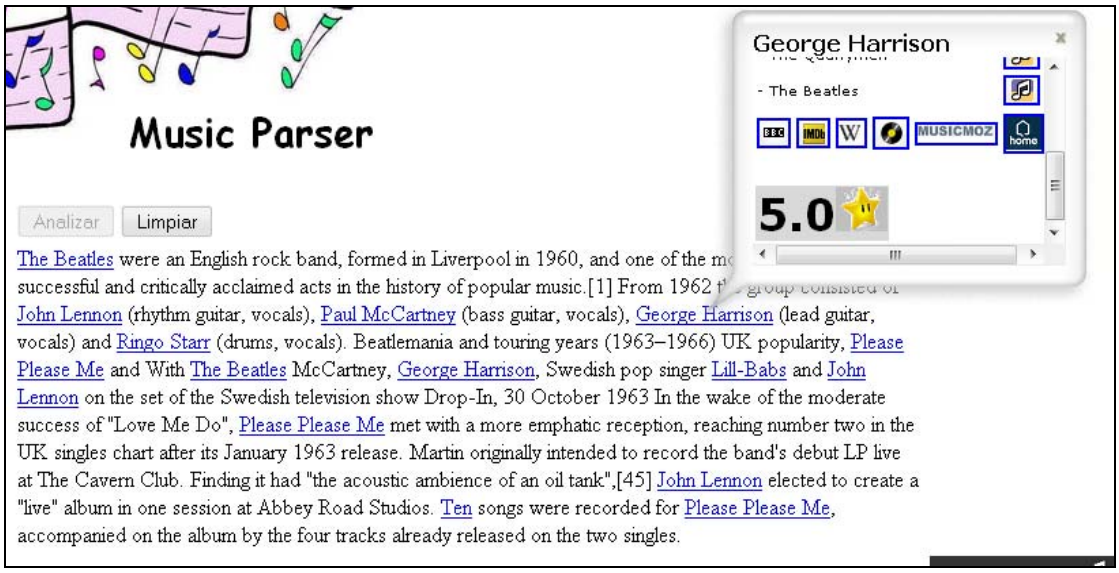


Ilustración 60: Resultado Prueba 01. Contenido del tooltip para Artista continuación

Se observa que el contenido es correcto, aparece información coherente en todos los apartados. Se han mostrado dos imágenes para analizar el contenido del mismo ya que se ha tenido que utilizar la barra de desplazamiento o scroll.

No se dispone de toda la información de los artistas en la base de datos de MusicBrainz, puede que el artista no haya tenido hijos o matrimonios que hayan estado relacionados con la música, o que los usuarios no hayan dado una valoración del mismo. A continuación se comprueba que la información mostrada es coherente con esta posibilidad:

Ilustración 61: Resultado Prueba 01. Contenido del tooltip para Artista sin todo el contenido

El artista no tiene hijos artistas, con lo que no aparece esta información y no afecta al resto del contenido ni a la presentación del mismo. Tampoco hay valoración de los usuarios de este artista, con lo que no se muestra:

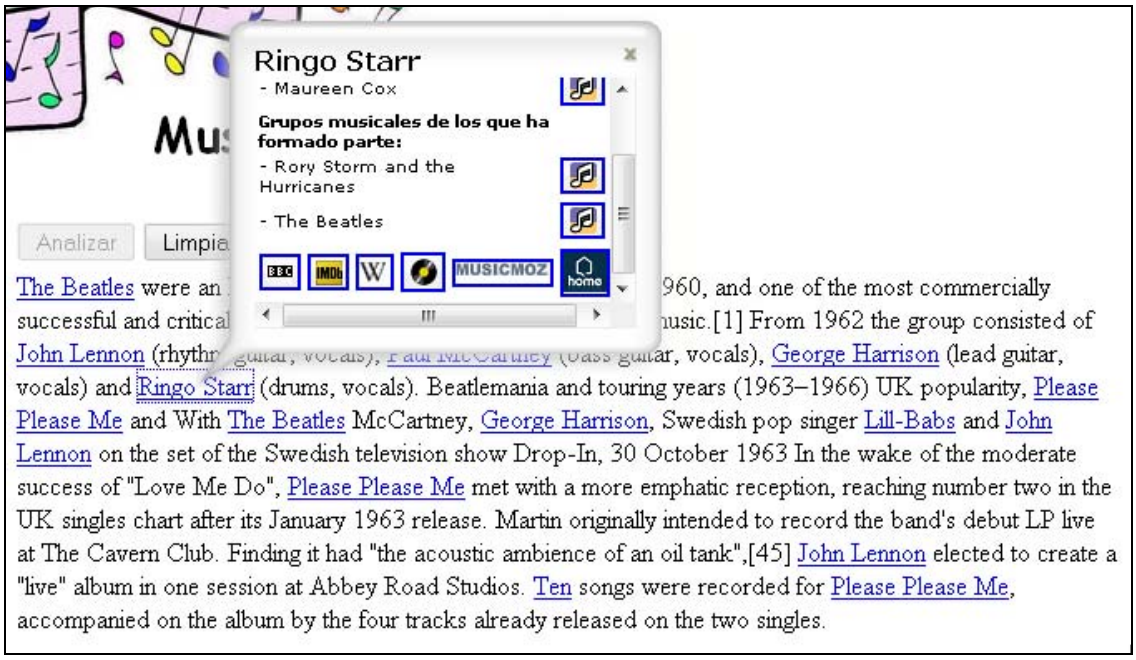


Ilustración 62: Resultado Prueba 01. Contenido del tooltip para Artista sin Rating

Se comprueba que el comportamiento es el equivalente para las entidades de tipo Grupo de música. En este caso mostrando los miembros del grupo, los álbumes del mismo, los enlaces relacionados y la valoración que hacen los usuarios de MusicBrainz del mismo:

Ilustración 63: Resultado Prueba 01. Contenido del tooltip para Grupo

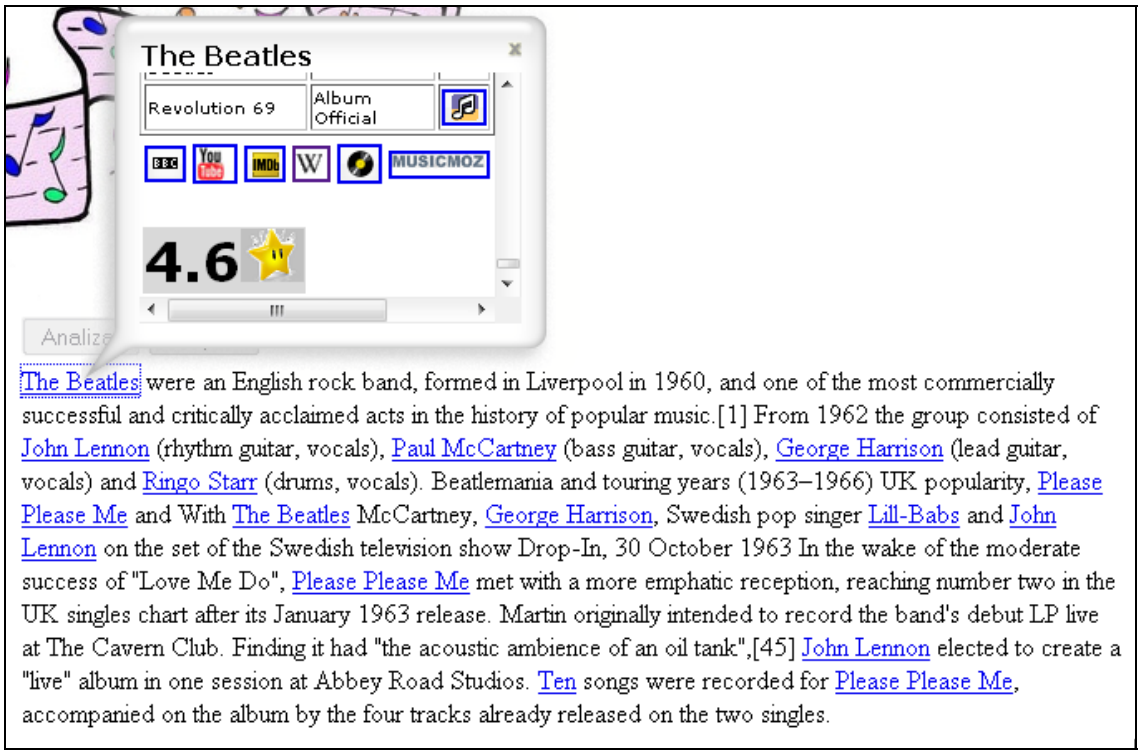


Ilustración 64: Resultado Prueba 01. Contenido del tooltip para Grupo continuación

Avanzando en la ventana auxiliar o tooltip con la barra de desplazamiento se comprueba que el contenido es el apropiado a este tipo de entidad y además es coherente.

Se comprueba ahora que sólo se muestra la información de la que se dispone, no hay información incoherente, para ello se va a aprovechar la inclusión de Ten como grupo musical, aunque realmente no se refiera a él este texto.

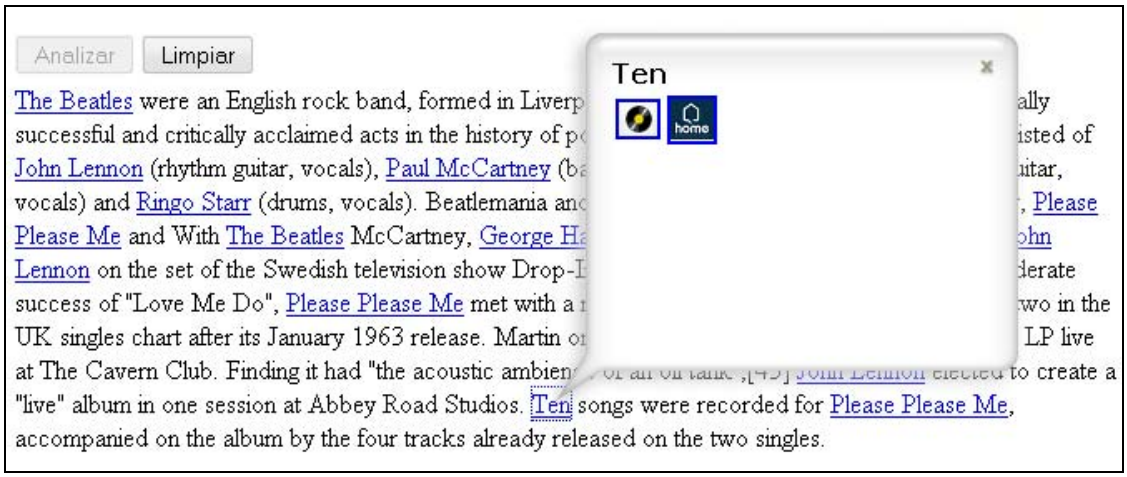


Ilustración 65: Resultado Prueba 01. Contenido del tooltip para Grupo sin toda la información

Efectivamente no se ha incluido referencia a ninguna información de la que no se tuviera información en la base de datos de MusicBrainz, sólo se muestra de la que se dispone, que en este caso es los enlaces relacionados.

Se comprueba también que si hay más de una repetición dentro del texto es capaz de mostrar la misma información en todas ellas:

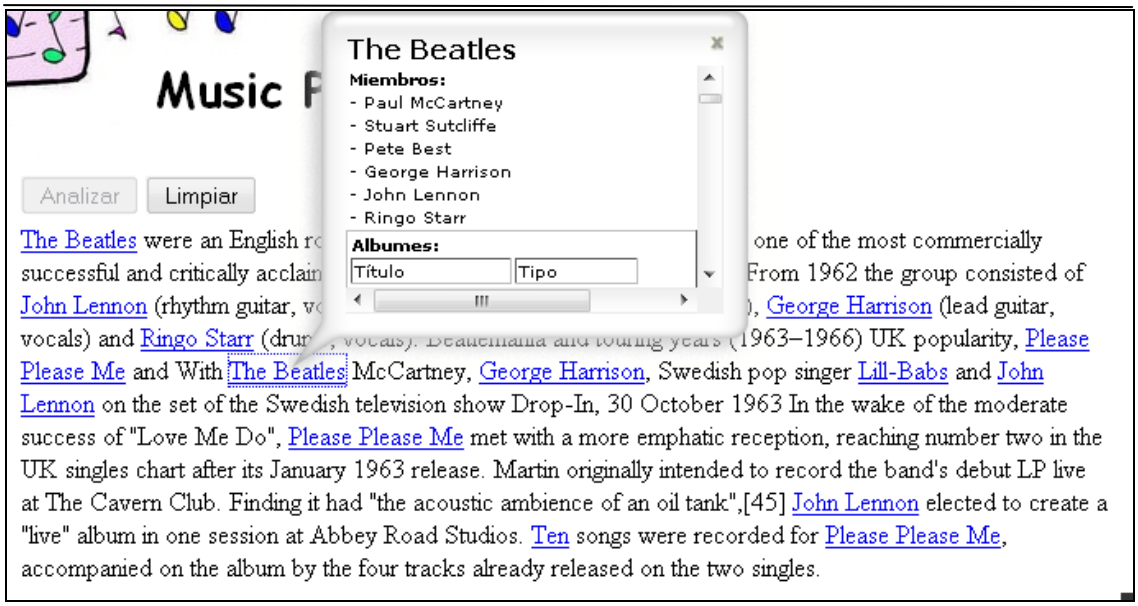


Ilustración 66: Resultado Prueba 01. Contenido del tooltip para Grupo en la repetición

Se comprueban ahora las entidades pertenecientes a la categoría Álbum de música, en las que tiene que aparecer información referente al autor, las pistas o canciones que lo componen, las versiones que han sacado al mercado del mismo, valoración por los usuarios y los enlaces relacionados:

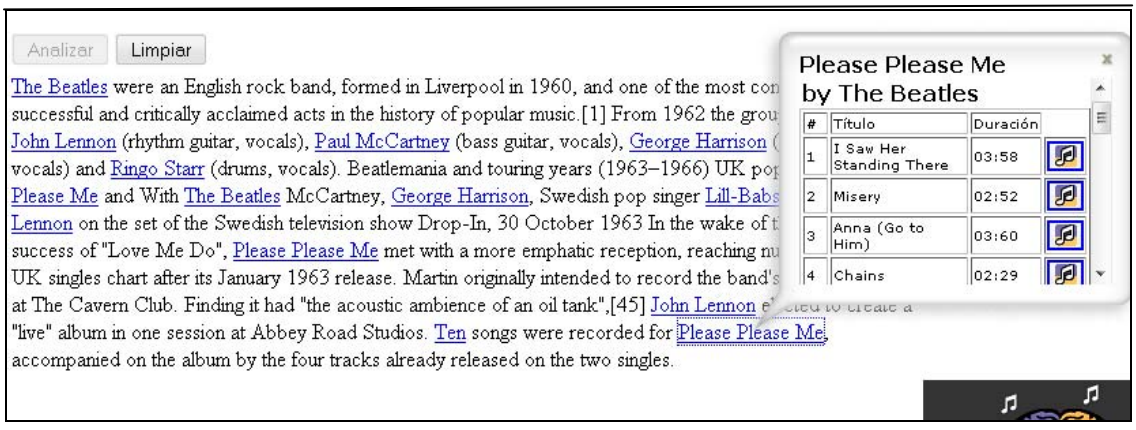


Ilustración 67: Resultado Prueba 01. Contenido del tooltip para Álbum

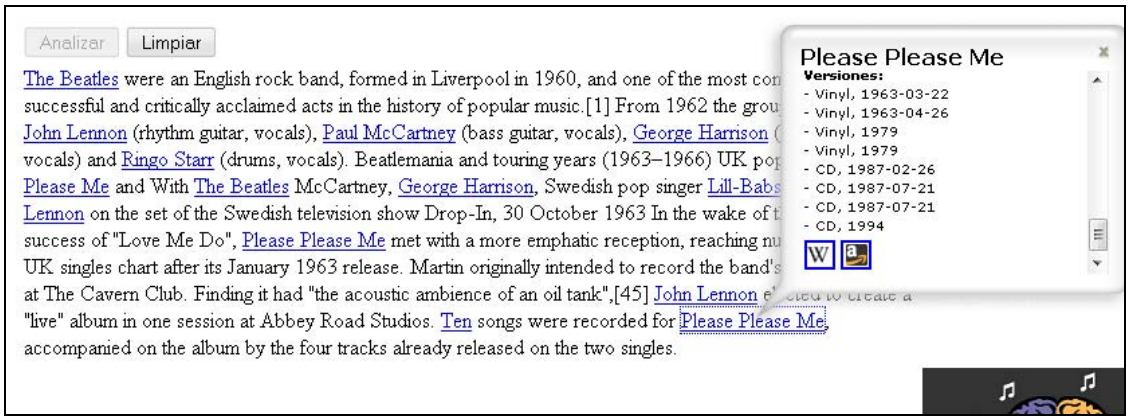


Ilustración 68: Resultado Prueba 01. Contenido del tooltip para Álbum continuación

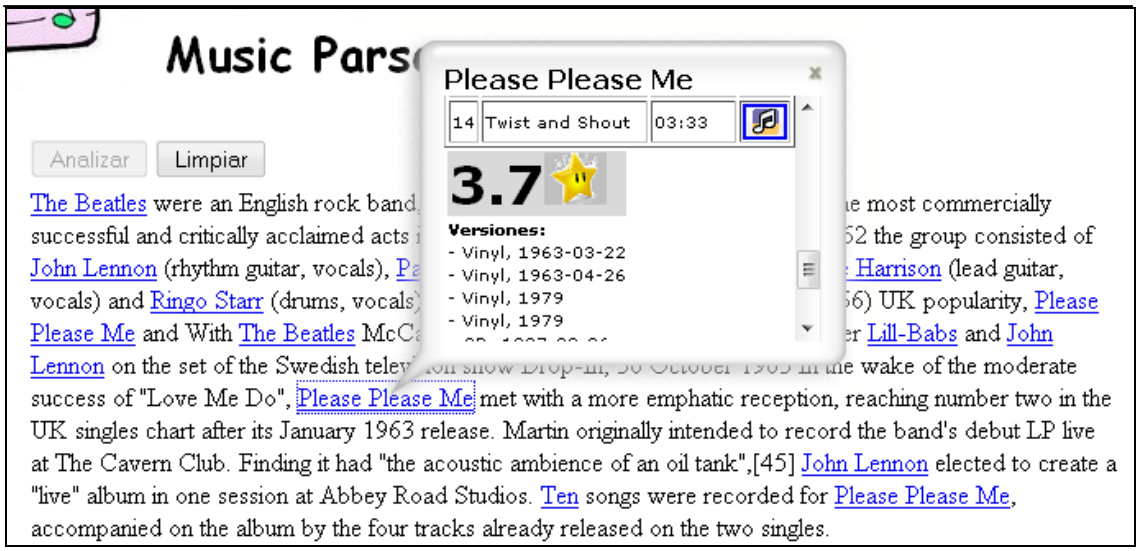


Ilustración 69: Resultado Prueba 01. Contenido del tooltip para Álbum continuación

La información es coherente. Además se comprueba que se corresponde con la mostrada para las otras repeticiones de la misma entidad dentro del texto.

Tiempo empleado:

- Llamada al servicio Web Calais: 3,85 segundos.
- Media de las llamadas a MusicBrainz: 1,37 segundos
- Repeticiones por errores: 3.
- Total: 1 minuto y 16 segundos.

Conclusiones:

Es capaz de diferenciar entre los diferentes tipos de entidades, aportando la información correcta en cada caso. Además se mantiene la coherencia de la información mostrada cuando la base de datos de MusicBrainz no dispone de algunos de los campos.

También es capaz de tratar con las repeticiones de una misma entidad, manteniendo la misma información en cada una de ellas.

Respecto a los tiempos de respuesta, se han reconocido 14 entidades, de cada una de ellas como mínimo se han realizado 2 llamadas a MusicBrainz, una llamada de búsqueda por nombre y la otra de búsqueda por MBID, por lo que el mínimo de tiempo empleado en las llamadas a MusicBrainz es de: 38,36 segundos. Añadiendo los 4 segundos por las repeticiones de las llamadas por errores, son 42,47 segundos, más los casi 4 que tarda en hacer la llamada inicial a Open Calais, hacen un total del 46,32 segundos gastados en llamadas a los servicios externos. Lo que deja un total de unos 20 segundos para el procesamiento de los textos por parte del sistema, un poco más de un segundo de procesamiento por cada entidad.

Son tiempos de respuesta aceptables, si además se tiene en cuenta que el transporte es lo que más consume y que eso no depende del sistema, si no de los servicios externos.

Prueba de uso limitado a una sola categoría: Texto perteneciente al contexto musical con solo una categoría de entidades. El texto ha sido extraído de Wikipedia.

Objetivos:

- Comprobar que es capaz de reconocer una sola categoría de entidades de manera consistente.
- Comprobar el tiempo empleado por el sistema en completar la solución.

Texto:

One of the most commercially successful and critically acclaimed acts in the history of popular music.[1] From 1962 the group consisted of John Lennon (rhythm guitar, vocals), Paul McCartney (bass guitar, vocals), George Harrison (lead guitar, vocals) and Ringo Starr (drums, vocals). Rooted in skiffle and 1950s rock and roll, the group later worked in many genres ranging from folk rock to psychedelic pop, often incorporating classical and other elements in innovative ways. The nature of their enormous popularity, which first emerged as the "Beatlemania" fad, transformed as their songwriting grew in sophistication. The group came to be perceived as the embodiment of progressive ideals, seeing their influence extend into the social and cultural revolutions of the 1960s.

Resultado:

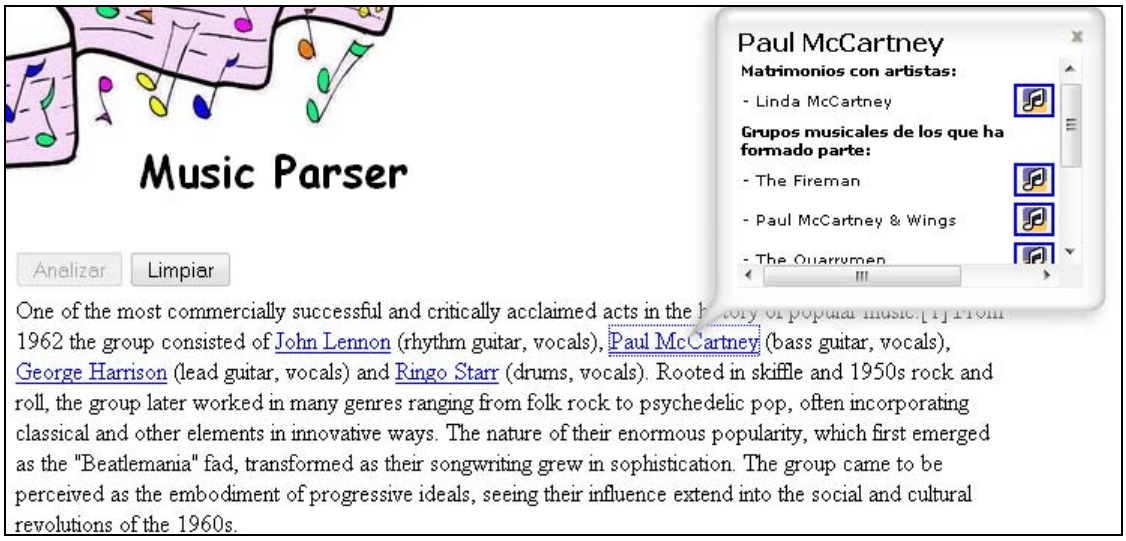


Ilustración 70: Resultado Prueba 02

No hay ningún error de reconocimiento, se reconocen todas las entidades que se tiene que reconocer, en este caso todas pertenecientes a la misma categoría, Artistas.

Tiempo empleado:

- Llamada al servicio Web Calais: 2,9 segundos.
- Media de las llamadas a MusicBrainz: 1,29 segundos
- Repeticiones por errores: 2
- Total: 21 segundos.

Conclusiones:

Es capaz de reconocer una única categoría de entidad sin errores y de mostrar la información correspondiente a la misma.

Respecto a los tiempos de respuesta, el transporte supone 15,8 segundos del total, ya que son necesarias dos llamadas como mínimo a MusicBrainz por cada entidad, más las dos por los errores. A esto hay que sumarle el tiempo invertido en la llamada a Open Calais. Con lo que supone 6,2 segundos de procesamiento por parte del sistema, un poco más de un segundo por cada entidad, lo que es coherente con los resultados obtenidos en la prueba1.

Por tanto que tarde más o menos tiempo, va a depender del número de llamadas erróneas a MusicBrainz, ya que por ahora los tiempos de llamada a MusicBrainz y los de procesamiento son muy parecidos. Los de Calais dependerán del tamaño del texto y del tamaño de la respuesta JSON.

Prueba de uso limitado a dos categorías: Texto perteneciente al contexto musical con dos categorías de entidades, que comparten una entidad con el mismo nombre pero distinta categoría, es decir, el nombre del grupo y el disco coinciden. El texto ha sido extraído de Wikipedia.

Objetivos:

- Comprobar que es capaz de reconocer dos categorías de entidades de manera consistente.
- Comprobar que el sistema guarda la coherencia semántica entre diferentes categorías.
- Comprobar el tiempo empleado por el sistema en completar la solución.

Texto:

Train is an American rock band from San Francisco, California, formed in 1994. The group named Train achieved mainstream success with their debut album, Train, which was released in 1998. Their second album, Drops of Jupiter (2001) brought the band massive popularity. The lead single from the album, Drops of Jupiter was an international hit and won two Grammy Awards in 2002. The album was certified double platinum in the United States and Canada and remains the band's best-selling album to da

Resultado:

Ilustración 71: Resultado Prueba 03

Se han reconocido todas las entidades que había en el texto, así como sus repeticiones. Si vemos el contexto en el que se encuentran las entidades vemos que una de las repeticiones de la entidad “Train” se refiere a una entidad de la categoría Álbumes musicales y las otras dos a la categoría Grupos musicales. Calais ha sido capaz de desambiguar estas dos entidades y las ha colocado en cada una de sus categorías, ahora hay que comprobar que el sistema mantiene esta semántica.

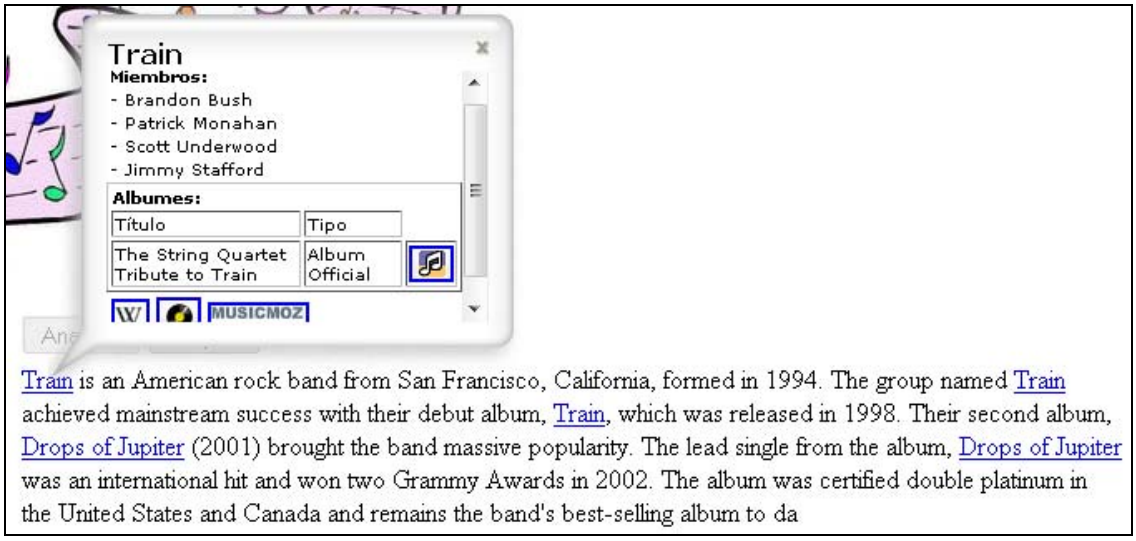


Ilustración 72: Resultado Prueba 03. Contenido Grupo

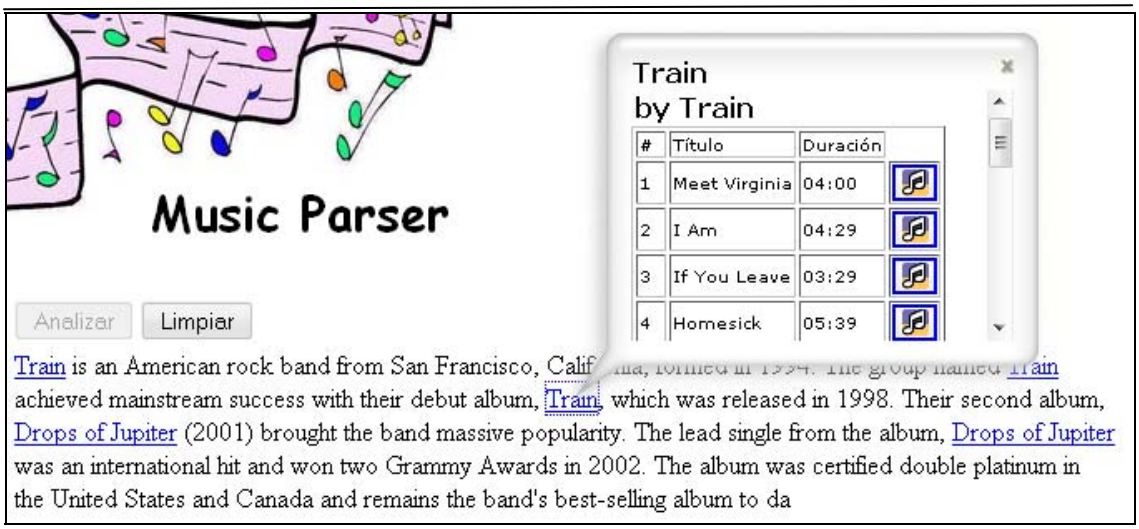


Ilustración 73: Resultado Prueba 03. Contenido Álbum

Efectivamente distingue entre los tipos de entidades y muestra para cada una de ellas la información adecuada.

Tiempo empleado:

- Llamada al servicio Web Calais: 3,29 segundos.
- Media de las llamadas a MusicBrainz: 1,12 segundos
- Repeticiones por errores: 0
- Total: 19 segundos.

Conclusiones:

El sistema es capaz de responder sin errores a un texto en el que sólo aparezcan dos tipos de entidades.

Además mantiene la semántica que aporta Calais siendo capaz de desambiguar términos dependiendo del contexto.

Respecto a los tiempos coincide con lo obtenido en las pruebas 1 y 2, si se tiene en cuenta que el tiempo de las llamadas a MusicBrainz ha disminuido. Esto puede ser debido a que se dispone de menos información en la base de datos de MusicBrainz a cerca del grupo Train, que de Los Beatles, como cabe esperar. En esta ocasión se ha empleado un poco más de 14 segundos en transportes, y un segundo de procesamiento de las entidades.

Prueba de ambigüedad – contexto 1: Texto perteneciente al contexto musical que contiene al menos un acrónimo y un nombre que pueda resultar ambiguo. El texto ha sido extraído de Wikipedia.

Objetivos:

- Comprobar si es capaz de reconocer los acrónimos.
- Comprobar si es capaz de romper la ambigüedad en determinados contextos.

Texto:

While a few artists like R.E.M. and The Cure achieved commercial success and mainstream critical recognition, many alternative rock artists during the 1980s were cult acts that recorded on independent labels and received their exposure through college radio airplay and word-of-mouth. With the breakthrough of Nirvana and the popularity of the grunge and Britpop movements in the 1990s, alternative rock entered the musical mainstream and many alternative bands became commercially successful.

Resultado:

El servicio Web de OpenCalais debería haber reconocido como entidades R.E.M., The Cure y Nirvana, sin embargo sólo es capaz de reconocer R.E.M, el resto no es capaz de desambiguarlas con el contexto dado.

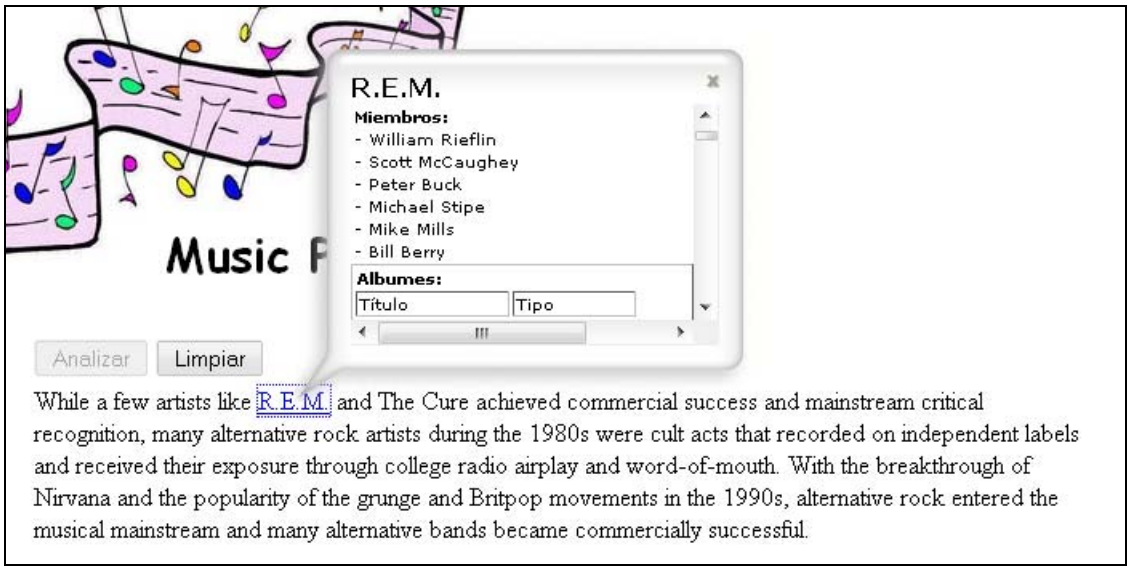


Ilustración 74: Resultado Prueba 04

Conclusiones:

De momento se ha demostrado que el contexto es importante a la hora de que OpenCalais sea capaz de reconocer determinadas entidades ya que pueden dar lugar a confusiones al tener otros significados en otros contextos. Este es el caso de las entidades que aparecen en este texto y el sistema está supeditado a lo que sea capaz de reconocer el servicio Web de Open Calais.

Prueba de ambigüedad – contexto 2: Texto perteneciente al contexto musical que contiene un nombre ambiguo que no ha sido capaz de reconocer en un contexto determinado, ahora en uno diferente. El texto ha sido extraído de Wikipedia.

Objetivos:

- Comprobar si es capaz de romper la ambigüedad en determinados contextos.

Texto:

Nirvana was an American rock band that was formed by singer/guitarist Kurt Cobain and bassist Krist Novoselic in Aberdeen, Washington in 1987. Nirvana went through a succession of drummers, the longest-lasting being Dave Grohl, who joined the band in 1990.

Resultado:

Una de las entidades que dentro del otro texto no ha sido capaz de ser reconocida, en otro contexto sí que lo es. Como en este texto que es más específico de la entidad Nirvana que OpenCalais no ha podido reconocer en el texto de la prueba 4. Sin embargo se puede observar

aquí el caso contrarío, la entidad Washington aparece como entidad musical reconocida, cuando en este contexto no se refiere al grupo musical si no a la ciudad.

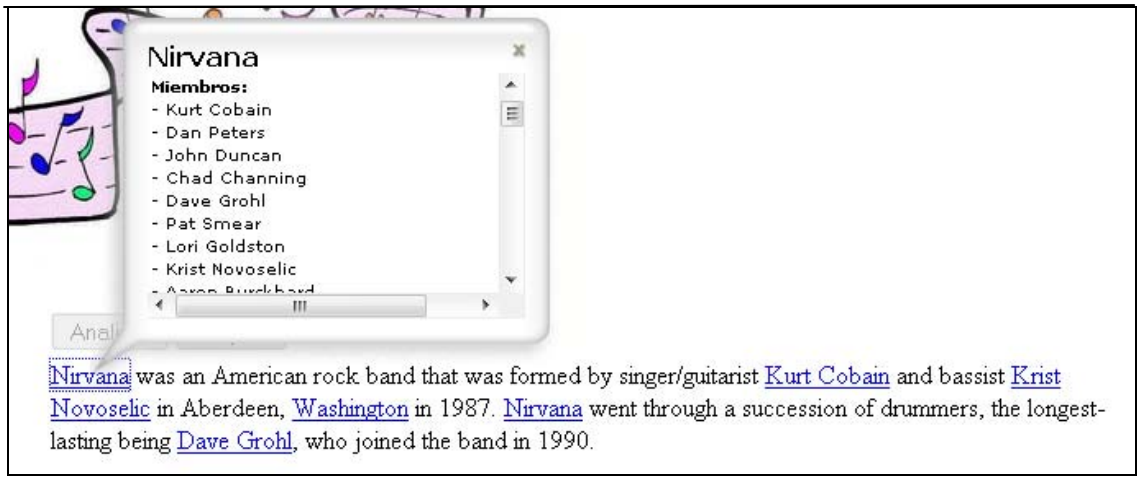


Ilustración 75: Resultado Prueba 05

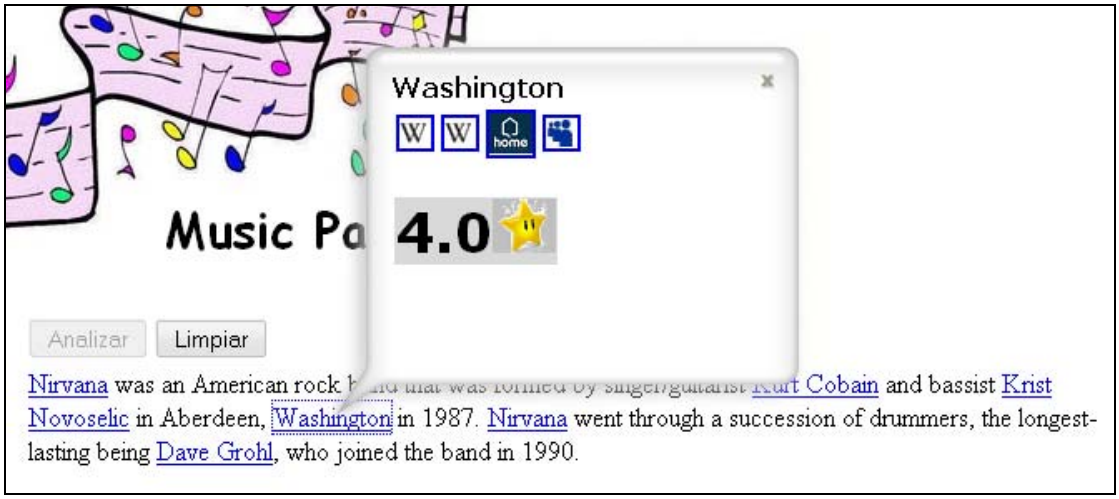


Ilustración 76: Resultado Prueba 05. Entidad mal reconocida

Conclusiones:

Se ha comprobado que el contexto influye en la respuesta que da el servicio Web de Open Calais, una entidad que no se ha reconocido en un texto si lo ha sido en otro más específico.

Ha salido a la luz una limitación en estas pruebas que es la dependencia de OpenCalais que hay veces que no es capaz de desambiguar bien las entidades. Esta limitación se ha observado en sus dos posibilidades, la de que no reconozca una entidad que debería reconocer y la contraría, que reconozca una que no debiera.

Prueba de texto procedente de otra fuente: Texto perteneciente al contexto musical no perteneciente a la Wikipedia. Ha sido extraído del sitio BBC Music.

Objetivos:

- Comprobar que es capaz de reconocer texto procedente de diversas fuentes.

Texto:

The Pixies are an American alternative rock band that formed in Boston in 1986. The group disbanded in 1993 under acrimonious circumstances but reunited in 2004. Black Francis (lead vocals and electric guitar and acoustic guitar), Joey Santiago (electric guitar), Kim Deal (bass guitar and backing vocals), and David Lovering (drums) have been the band's continual members. While the Pixies found only modest success in their home country, they were significantly more successful in the United Kingdom and elsewhere in Europe, although never achieving mainstream success with their studio albums.

Resultado:

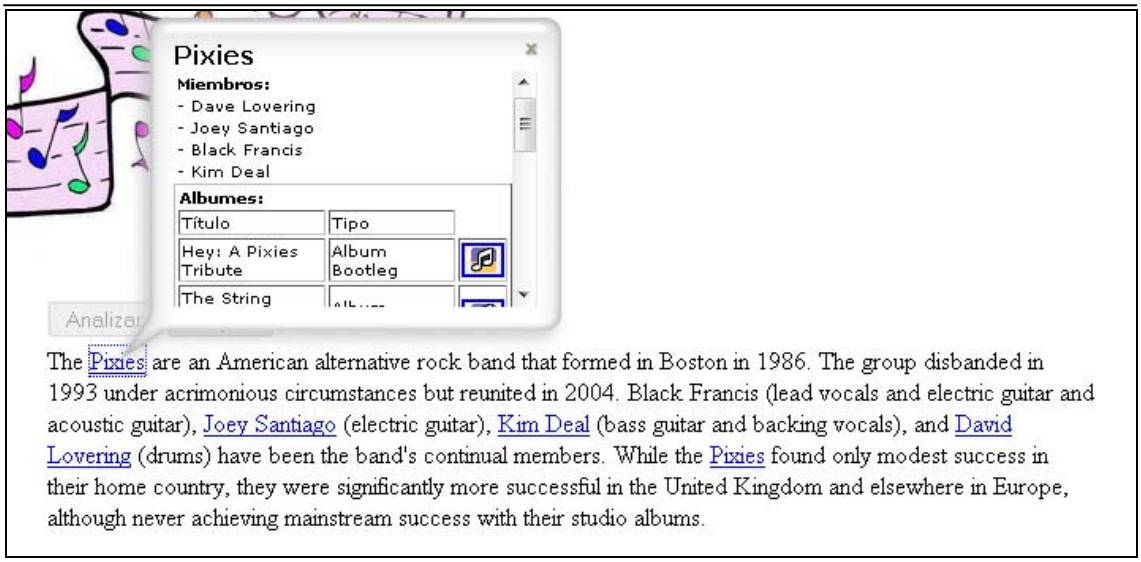


Ilustración 77: Resultado Prueba 06

Se han reconocido todas las entidades que se tenían que reconocer en este texto.

Conclusiones:

El sistema es capaz de reconocer texto que no sean exclusivos de la Wikipedia, que hasta ahora había sido la fuente principal para los textos de estas pruebas. Sí que es conveniente que se trate de un texto con contexto musical, como el elegido para esta prueba, para evitar los problemas de ambigüedad tratados en las pruebas 4 y 5.

Prueba de nombres incompletos: Texto perteneciente al contexto musical que tiene entidades en las que no aparece el nombre completo de las mismas.

Objetivos:

- Comprobar que es capaz de reconocer nombres incompletos cuando aportan la suficiente información por sí mismos.

Texto:

The Beatles were an English rock band, formed in Liverpool in 1960, and one of the most commercially successful and critically acclaimed acts in the history of popular music.[1] From 1962 the group consisted of John Lennon (rhythm guitar, vocals), Paul McCartney (bass guitar, vocals), George Harrison (lead guitar, vocals) and Ringo Starr (drums, vocals). Rooted in skiffle and 1950s rock and roll, the group later worked in many genres ranging from folk rock to psychedelic pop, often incorporating classical and other elements in innovative ways. The nature of their enormous popularity, which first emerged as the "Beatlemania" fad, transformed as their songwriting grew in sophistication. The group came to be perceived as the embodiment of progressive ideals, seeing their influence extend into the social and cultural revolutions of the 1960s.

With an early five-piece line-up of Lennon, McCartney, Harrison, Stuart Sutcliffe (bass) and Pete Best (drums), The Beatles built their reputation in Liverpool and Hamburg clubs over a three-year period from 1960.

Resultado:

Si observamos el texto de entrada, se observa que en el último párrafo se hace referencia a cada uno de los Beatles por su apellido sin incluir su nombre, sin embargo es capaz de reconocer que se trata de cada uno de los componentes del grupo.

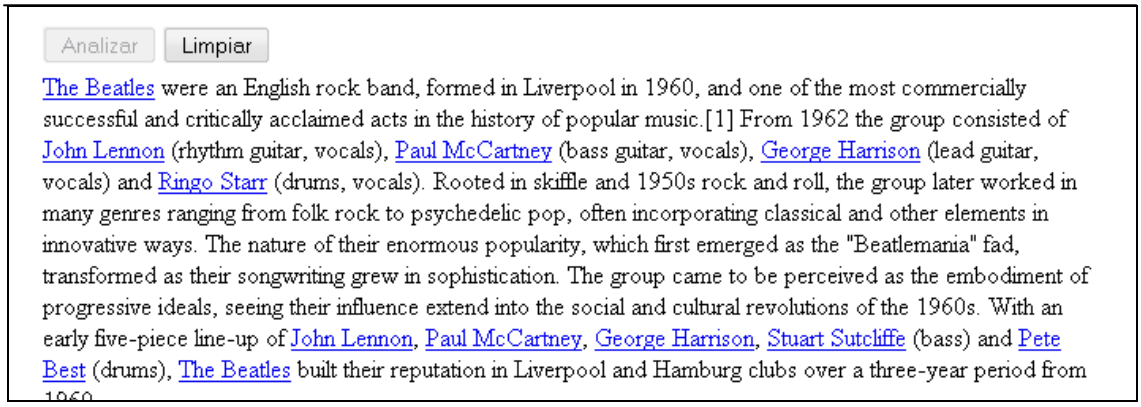


Ilustración 78: Resultado Prueba 07

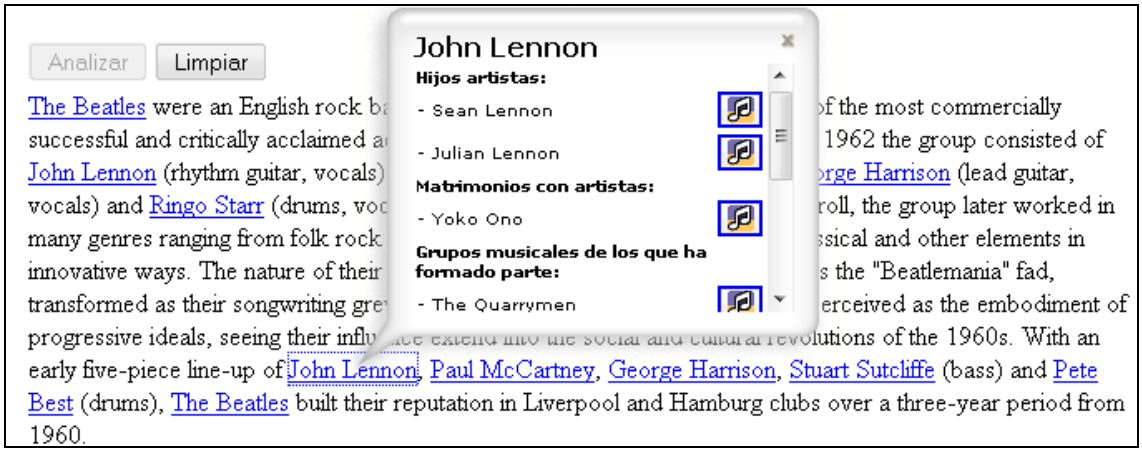


Ilustración 79: Resultado Prueba 07. Contenido Artista

Conclusiones:

El sistema ha sido capaz de reconocer esta vez a los componentes de los Beatles por sus apellidos. Esto es gracias a la respuesta que OpenCalais da, por la cual, si observamos el texto de entrada y el de salida, se ve que sustituye el apellido por el nombre completo. Esto es debido a que OpenCalais tiene la referencia de cada uno de ellos por su nombre completo y en la respuesta JSON hace referencia a los mismos de esta manera.

Prueba de nombres incompletos no reconocidos: Texto perteneciente al contexto musical que tiene entidades en las que no aparece el nombre completo de las mismas.

Objetivos:

- Comprobar que es capaz de reconocer nombres incompletos cuando no se aporta suficiente información en el texto que los acompaña.

Texto:

With an early five-piece line-up of Lennon, McCartney, Harrison, Stuart Sutcliffe (bass) and Pete Best (drums), The Beatles built their reputation in Liverpool and Hamburg clubs over a three-year period from 1960.

Resultado:

El texto de entrada se corresponde con el del último párrafo de la prueba 07, con la diferencia de que no se aporta el resto de la información de los otros párrafos. En el último párrafo se hace referencia a cada uno de los Beatles por su apellido sin incluir su nombre, y al no disponer de un contexto más amplio ni de más información no es capaz de reconocer estas entidades.

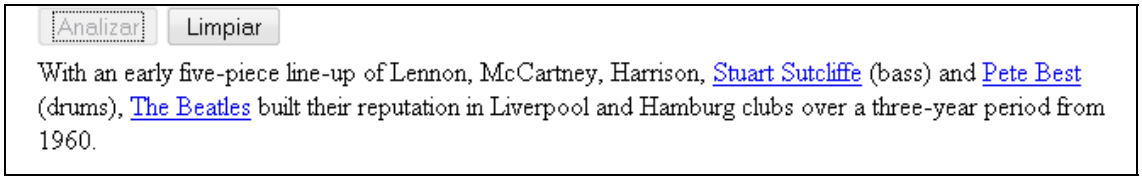


Ilustración 80: Resultado Prueba 08

Conclusiones:

El sistema no ha sido capaz de reconocer esta vez a los componentes de los Beatles por sus apellidos ya que no se disponía de la suficiente información en el texto que los acompañaba.

El mismo texto pero acompañado de más información sí lo reconoce porque ya se ha hecho referencia a estas entidades a través de su nombre completo. Por tanto el reconocimiento o no de entidades con nombres incompletos depende de la información adicional que se tenga de las mismas.

4.12. *Problemática encontrada y soluciones propuestas*

Durante el desarrollo del sistema y sobre todo durante la etapa de ejecución de las pruebas, se ha observado que el sistema se ve afectado por una serie de limitaciones. Estas limitaciones son heredadas de cada uno de los servicios Web externos que se utilizan. Al ser este un sistema dependiente de estos servicios Web externos, adopta estas limitaciones como propias.

Toda la información que aparece en este sistema, desde el texto analizado semánticamente hasta la información ampliada para cada entidad, proviene de Servicios Web externos y por tanto depende de las limitaciones de los mismos. Es decir, cualquier contenido, que debido a una carencia de los servicios Web, sea incorrecto no va a ser responsabilidad del sistema y no va a haber manera ni de evitar que se muestre ni de corregirlo en este sistema. El sistema tiene confianza ciega en los servicios Web externos utilizados y dará por hecho que la información recibida es correcta.

Cada uno de los servicios Web externos al sistema va a tener sus propias limitaciones y que va a traspasar al sistema. Por tanto la funcionalidad del sistema se va a ver afectada por el servicio Web externo sobre el que se apoye concretamente.

A continuación se detallan las limitaciones del sistema clasificándolas según la funcionalidad del sistema a la que afecta, ya que esta clasificación se corresponde también con la que se hace de los servicios Web externos utilizados en el sistema:

La funcionalidad de reconocimiento semántico de texto se va a ver afectada por las limitaciones que le imponga Open Calais, se han detectado las siguientes:

- Aunque el equipo de OpenCalais está trabajando en esta línea, actualmente existen limitaciones a la hora de reconocer abreviaturas, acrónimos y nombres incompletos.
  Esto se ha puesto de manifiesto al realizar las pruebas 04, 07 y 08, cuyo objetivo era precisamente comprobar si el sistema aceptaba entidades identificadas por este tipo de nombres.
  Para el caso concreto del sistema:
  - Acrónimos: son reconocidos si la entidad musical tiene como nombre registrado dentro del panorama musical este acrónimo, por ejemplo el grupo R.E.M. Aún así este tipo de entidades son más difíciles de reconocer y van a depender en mayor nivel del contexto que las acompañe.
  - Nombres incompletos: este tipo de entidades son más propensas a no ser reconocidas si el contexto que las acompaña no es muy específico, ya que no deja de ser información que no se le da al sistema. Por ejemplo en la prueba 07 ha reconocido a los Beatles por sus apellidos pero en la prueba 08 no lo ha hecho al tener un contexto menos específico y que aporta menos información.

- Debido a que Calais surgió expresamente para realizar reconocimientos semánticos de noticias, hay entidades de otros contextos que no termina de reconocer ya que su base de reglas debe ser más limitada para éstos. Esta limitación se hace más latente para este sistema a la hora de reconocer álbumes musicales, pero desapareciendo a medida que el servicio Web de OpenCalais vaya añadiendo nuevas entidades y ampliando los contextos.
  En este sistema se ha comprobado que ocurre en la prueba 04 que no termina de reconocer los grupos musicales que debería.

- Este sistema está limitado por las entidades que este servicio Web reconozca y éste hay en contextos en los que no detecta o no lo hace de manera correcta determinadas entidades. Ha quedado latente esta limitación en la prueba 05 por ejemplo, en la que ha reconocido Washington como entidad musical, cuando realmente se refería a la ciudad. También se ha observado esta característica en las pruebas 04 y 05 con la entidad Nirvana, observando cómo dependiendo del contexto es capaz de reconocer a determinadas entidades o no.

Ninguna de las limitaciones relativas a OpenCalais encontradas tiene solución, ya que en este sentido, en la funcionalidad que aporta Calais, el sistema depende totalmente de la respuesta que de éste se obtenga.

Por otro lado, la funcionalidad de ampliación de la información de las entidades musicales reconocidas en un texto, estará supeditada al servicio Web de MusicBrainz que es el que le aporta la información que el sistema necesita. A continuación se tratan estas limitaciones:

- La información que se desea mostrar de cada una de las entidades puede no estar disponible al completo en algunas de ellas, por ejemplo para artistas no muy conocidos la información disponible en la base de datos de MusicBrainz sobre los mismos puede ser reducida.
  Esto es lo que ha ocurrido para el grupo Ten en la prueba 01, al no ser un grupo muy seguido por los usuarios de MusicBrainz, se dispone de menos información acerca del mismo.
  Esta limitación va a seguir siéndolo ya que la información adicional de cada una de las entidades se consigue exclusivamente a través de las consultas realizadas a este servicio Web. Por tanto la calidad y cantidad de la información que el sistema va a mostrar de cada entidad va a estar limitada por la que tenga disponible este servicio Web.

- El servicio Web de MusicBrainz no permite hacer más de una llamada por segundo, por lo que el tiempo de procesado del texto completo va a tardar el tiempo correspondiente a un segundo por cada llamada y contando que cada entidad reconocida necesita de dos llamadas al servicio, la primera para encontrar su identificador y la segunda para obtener la información, por cada entidad tardará dos segundos, más lo que tarde el sistema en realizar el procesado de la respuesta. A esto hay qua añadirle el hecho de que a veces este servicio Web se encuentra ocupado y devuelve error , por lo que habría que esperar otro segundo y volver a realizar la llamada.
  En las pruebas 01,02 y 03, en el apartado de la toma de tiempos aparecen reflejadas las repeticiones que ha habido que realizar de determinadas llamadas, debido a esta limitación. Por tanto esta limitación no afecta a la funcionalidad, pero sí al tiempo de respuesta del sistema.
  Esta limitación se ha resuelto mediante una función que duerme al sistema durante un segundo tras hacer cada una de las llamadas a MusicBrainz, de esta manera se asegura que no se realice más de una por segundo. Además para el caso de que el sistema se encuentre ocupado por estar atendiendo muchas llamadas de otros usuarios, el sistema se mantiene a la espera de que devuelva una respuesta de éxito.

5. Conclusiones

5.1. *Conclusiones*

Tras la finalización del proyecto se puede concluir que éste ha cumplido con los objetivos propuesto al inicio del mismo.

Resumiendo los objetivos planeados al inicio del proyecto y agrupándolos en dos grandes categorías: Objetivos personales del proyecto y Objetivos funcionales del proyecto.

Dentro de los llamados objetivos personales del proyecto, estaba la necesidad de entrar en contacto con el concepto de Web semántica y la de ampliar conocimientos relativos a los servicios Web así como de la realización de integraciones con los mismos. Estos objetivos han sido satisfechos.

En primer lugar se ha realizado el estudio requerido sobre los conceptos y tecnologías relativas a los servicios Web y a la Web semántica.

A raíz de las conclusiones de estos estudios se ha podido determinar cuál era la mejor manera de abordar el problema que se deseaba resolver. Es decir el la creación de un sistema que estuviera integrado con un servicio Web semántico y con un servicio Web contenedor de información musical.

Para poder llevar a cabo esta integración se han tenido que realizar una serie de formaciones en determinadas tecnologías que nunca antes había utilizado, tales como: Ajax, PHP, DOM o JSON. Estas tecnologías son de gran importancia actualidad, por lo que ha resultado muy útil haber profundizado en ellas. A estos conocimientos adquiridos se les añade el hecho de haber realizado un proyecto real de principio a fin, pasando por todas las etapas del ciclo de vida del Software, desde el análisis inicial del problema a través de la toma de requisitos hasta la construcción de un manual de usuario final.

Respecto a los objetivos funcionales del sistema. El primero era reconocer semánticamente un texto del contexto musical, y el segundo de ellos era el de dotar al texto de la capacidad de ampliar la información relativa a determinadas entidades procedentes del análisis semántico.

Se puede concluir que también han sido satisfechos, ya que se ha construido un sistema capaz de llevar a cabo el reconocimiento semántico de un texto, de seleccionar con esta información semántica aquellos términos relacionados con el ámbito musical y de ampliar la información de estos términos mostrando información de manera legible para el usuario y en un formato vistoso.

Analizando cada uno de estos objetivos en detalle, respecto al análisis semántico de los textos. Se ha conseguido el objetivo personal de entrar en contacto con el mundo de la Web semántica y con los conceptos que la acompañan. Como este análisis se ha conseguido con la ayuda de un servicio Web externo, se ha conseguido cumplir aquí el segundo objetivo personal, el de ser capaz de integrar un sistema con un servicio Web externo.

Respecto al objetivo funcional relativo a la ampliación de la información, se ha llevado a cabo con la ayuda otro servicio Web externo, por lo tanto se ha cumplido también el objetivo personal de indagar en la integración de servicios Web.

Por tanto se ha construido un sistema que ha cumplido tanto con los objetivos personales como con los funcionales. Además, como se verá a continuación se ha construido un sistema ampliable en un futuro, que es muy difícil que quede obsoleto.

5.2. *Futuras líneas de trabajo*

Este sistema, al estar desarrollado basándose en conceptos de última generación y al utilizar tecnología fácilmente reutilizable y actual, tiene la posibilidad de seguir desarrollándose en un futuro, ya sea ampliando su funcionalidad o facilitando el acceso y la distribución al usuario.

Respecto a futuras ampliaciones en su funcionalidad se pueden considerar las siguientes:

- Se podría ampliar el número de categorías a reconocer dentro de los textos. Conforme OpenCalais se vaya desarrollando y ampliando sus categorías e información de las mismas, el número de categorías podría crecer. Por ejemplo empezar a reconocer: eventos relacionados con los discos, como lanzamientos o conciertos, títulos de canciones, sellos musicales, etc.

- Se podría ampliar la información mostrada sobre las entidades. Dependiendo de la categoría a la que pertenezcan, se podrían realizar consultas a otros servicios Web externos también relacionados con la música. Por ejemplo, los servicios Web Last.fm, fullritmo, sawgi o groone que permiten escuchar música en vivo y compartirla a través de las redes sociales.

Para mejorar la distribución del sistema y facilitar al usuario el acceso al mismo:

- Se podría convertir en un add-on para Mozilla Firefox, ya que al estar su funcionalidad desarrollada completamente en JavaScript podría integrarse, ya que estos add-ons están formados por código JavaScript y código XUL para la parte visual.

6. Referencias Bibliográficas y material consultado

1. **Santiago Márquez Solís.** *La Web Semántica.* Santiago Márquez Solís, 2009.
2. **Gustavo Alonso, Fabio Casati, Harumi Kuno, Vijay Machiraju.** *Web services: concepts, architectures and applications.* Springer Verlag, 2004.
3. **Michael P. Papazoglou.** *Web Services: principles and technology.* Prentice Hall, 2007.
4. **Leonard Richardson, Sam Ruby.** *RESTful web services.* O'Reilly Media, 2007.
5. **MSD Magazine–More On REST:** http://msdn.microsoft.com/es-es/magazine/dd942839.aspx#id0070002 . Accedido en Mayo 2010.
6. **Sitio oficial de W3C:** http://www.w3c.es/ . Accedido en Abril y Mayo del 2010.
7. **Wikipedia:** http://en.wikipedia.org/ . Accedido desde Febrero hasta Junio del 2010.
8. **Sitio oficial del servicio Web de MusicBrainz:**

   http://musicbrainz.org/doc/XML_Web_Service . Accedido desde Febrero hasta Junio del 2010.
9. **Sitio oficial del servicio Web de OpenCalais:**

   http://www.opencalais.com/documentation/calais-web-service-api . Accedido desde Febrero hasta Junio del 2010.
10. **Sitio oficial de PHP:** http://php.net/ . Accedido en Mayo de 2010.
11. **Introducción a JSON:** http://www.json.org/json-es.html. Accedido en Marzo y Abril del 2010.
12. **Curso de Ajax y PHP:**

    http://illasaron.com/upload/search_result.php?query=ajax&search=Search . Accedido en Marzo y Abril del 2010.

7. Anexos

7.1. *Anexo I: Manual de usuario*

Software necesario:

- Servidor Apache 2.2 (http://httpd.apache.org/download.cgi)

- PHP versión 5 (http://php.net/downloads.php)

Para el entorno de pruebas, por comodidad, se decidió utilizar una solución paquetizada, llamada WampServer en su versión 2.0i (07/11/09) que incluye:

- Apache 2.2.11
- MySQL 5.1.36
- PHP 5.3.0

(http://www.wampserver.com/en/download.php)

Puesta en marcha del sistema:

1. Coger el fichero .zip con la solución (calais.zip) y descomprimir en el directorio deseado.

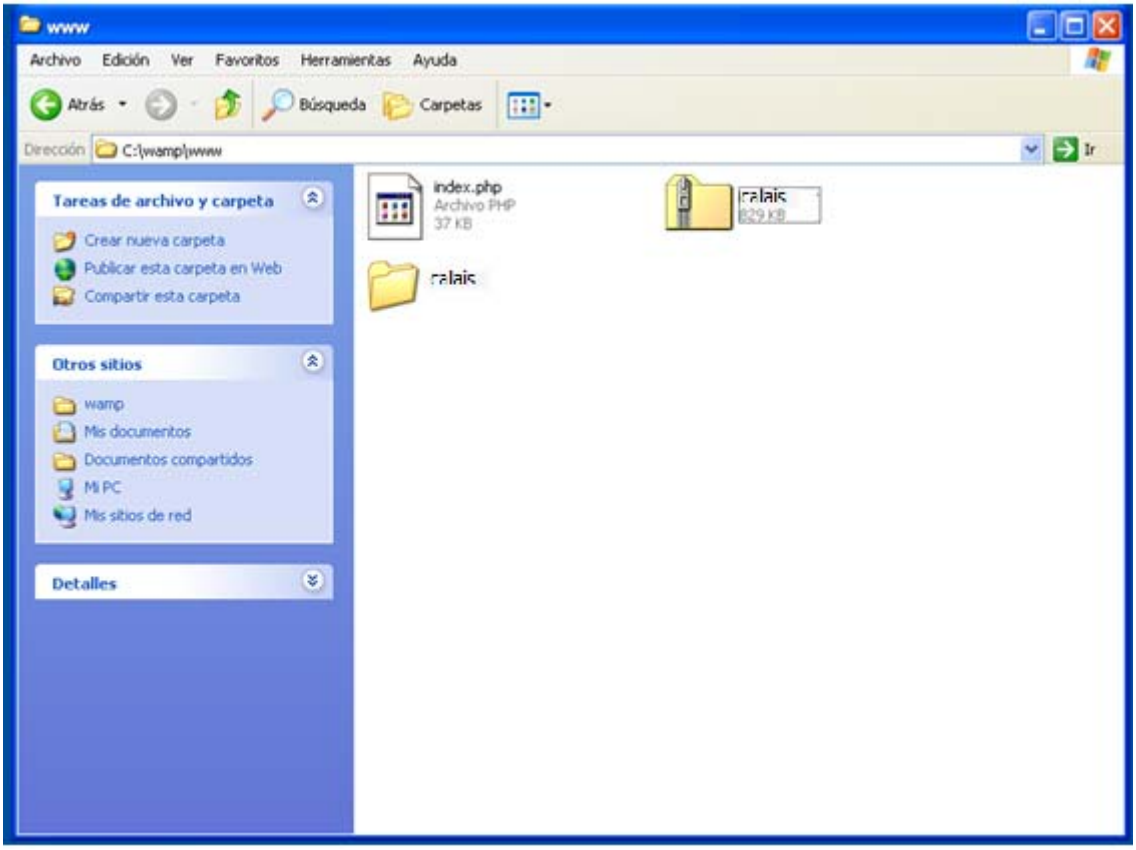


Ilustración 81: Puesta en marcha. Descomprimir ZIP

2. Tanto si se utiliza Apache por separado como el Apache integrado en WampServer hay que configurar el fichero httpd.config:

a. Con Apache simple:

Seleccionar en Programas de Windows el Apache http Server 2.2, Configure Apache Server, Edit the Apache httpd.conf Configuration File:

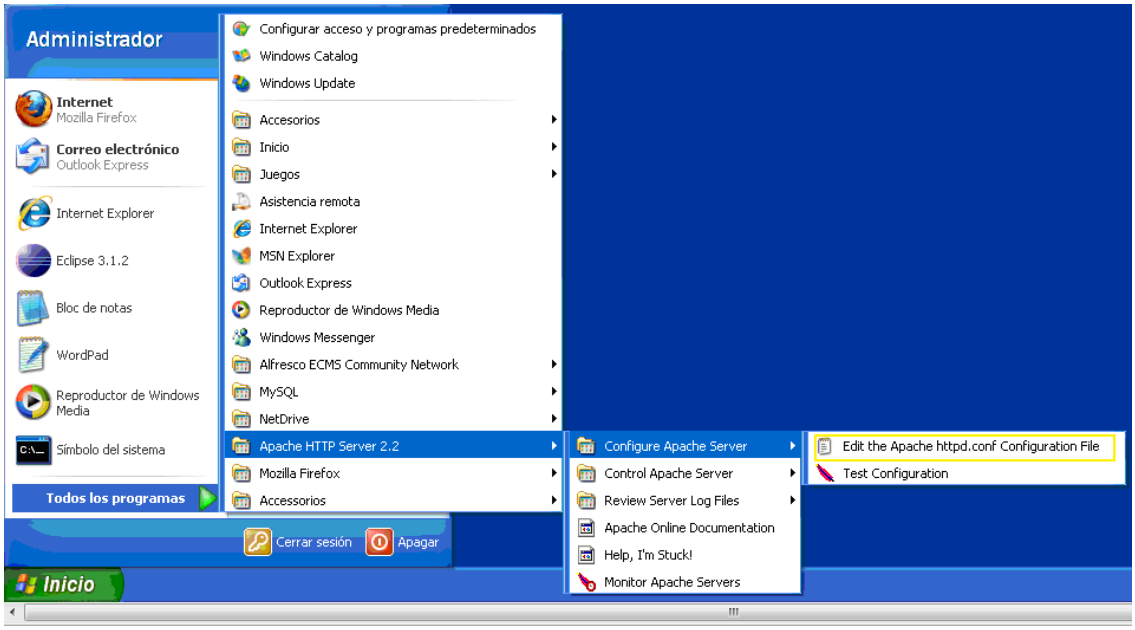


Ilustración 82: Puesta en marcha. Editar fichero configuración de Apache

b. Con WampServer:

Aparecerá un icono de acceso rápido en la barra del reloj:

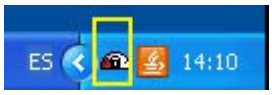


Ilustración 83: Puesta en marcha. Acceso a WampServer

Pulsar sobre este icono con el ratón y aparecerá el siguiente menú, del que hay que seleccionar el httpd.conf:

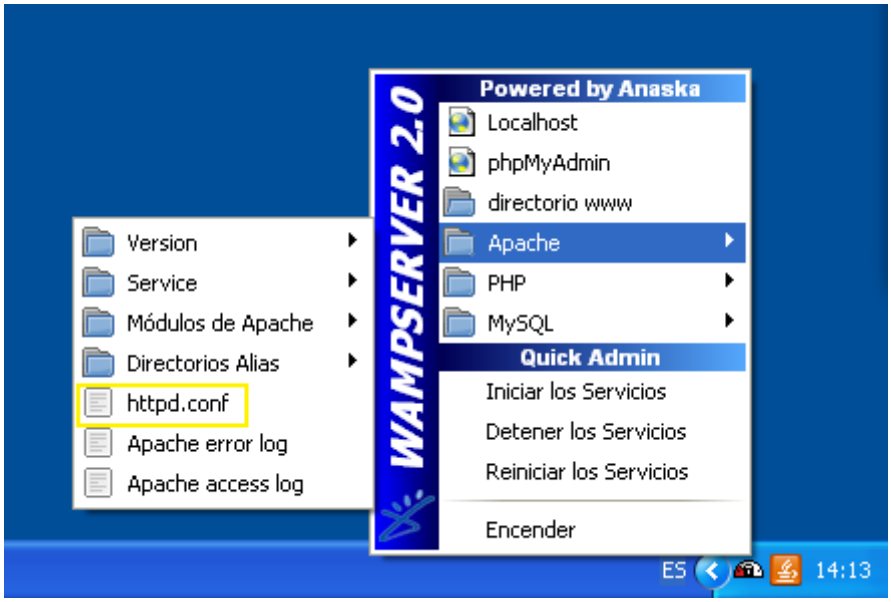


Ilustración 84: Puesta en marcha. Editar configuración de Apache con WampServer

c. Para ambos:

Del httpd.conf Configuration File hay que modificar el fragmento de la ruta de los documentos, e indicar dónde se ha descomprimido la solución (calais.zip):

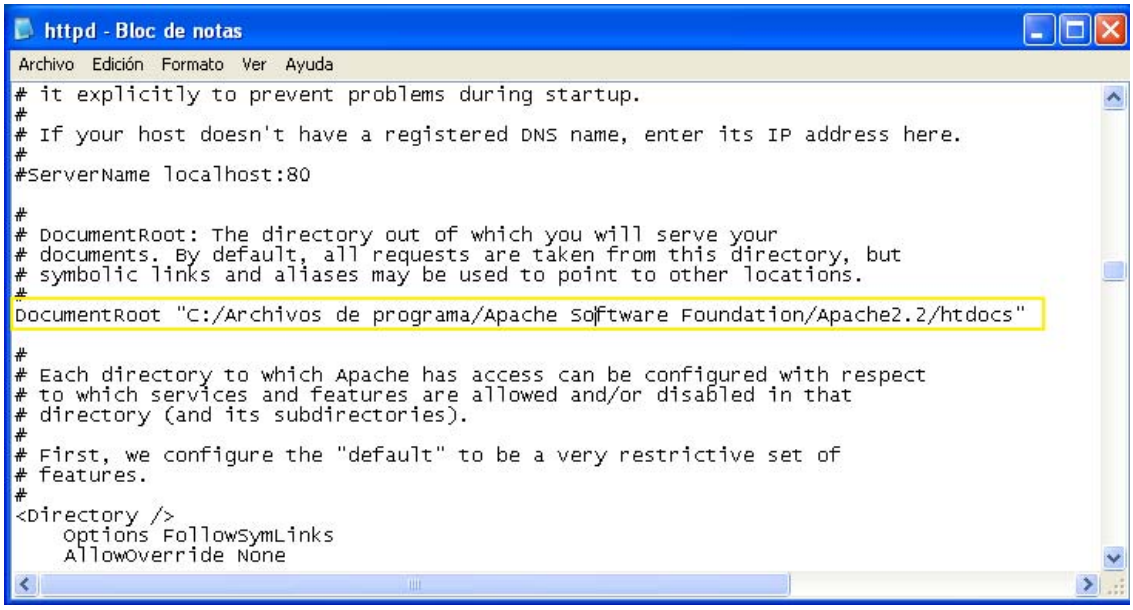


Ilustración 85: Puesta en marcha. Información del fichero de configuración de Apache

La que aparece en remarcada es la ruta que viene por defecto, se puede comentar añadiendo '#' a la línea y añadir la nueva:

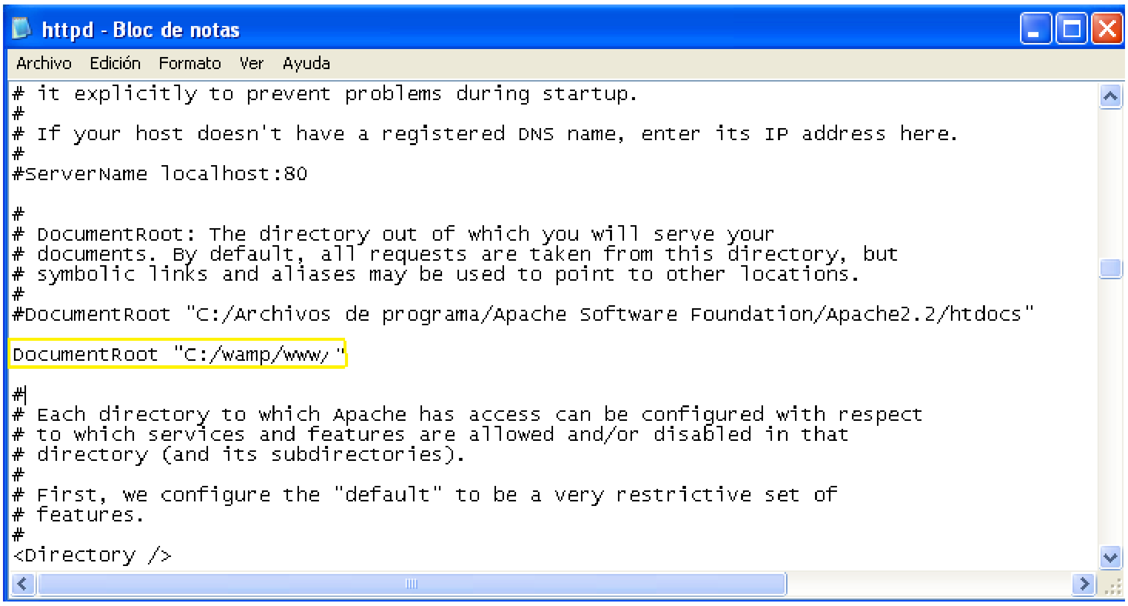


Ilustración 86: Puesta en marcha. Información del fichero de configuración de Apache

3. Ahora hay que levantar el servidor.

   En la esquina inferior derecha en el menú de inicio rápido, aparecerá un icono correspondiente al servidor Apache, que aparece con un cuadrado rojo, indicando que el servidor está parado:

Ilustración 87: Puesta en marcha. Icono Apache

Pulsar sobre este icono con el botón izquierdo para que aparezca el siguiente menú:

Ilustración 88: Puesta en marcha. Iniciar Apache

O sobre el derecho y seleccionar la opción Open Apache Monitor para que aparezca la consola completa:

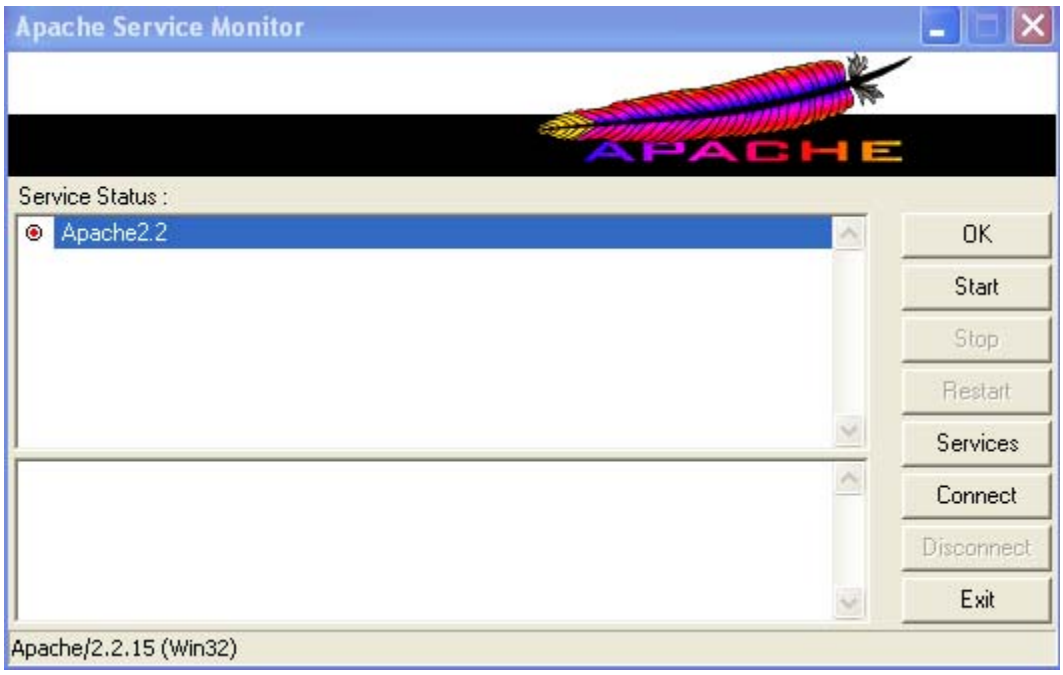


Ilustración 89: Puesta en marcha. Iniciar Apache

En cualquiera de los casos, pulsar **Start** para arrancar el servidor.

Si no ha ocurrido ningún problema durante el arranque, el icono aparecerá ahora con un triángulo en verde:

Ilustración 90: Puesta en marcha. Icono Apache iniciado

4. Una vez hecho esto se puede acceder a la página. En el navegador tecleamos http://localhost/calais y esto es lo que nos encontramos:

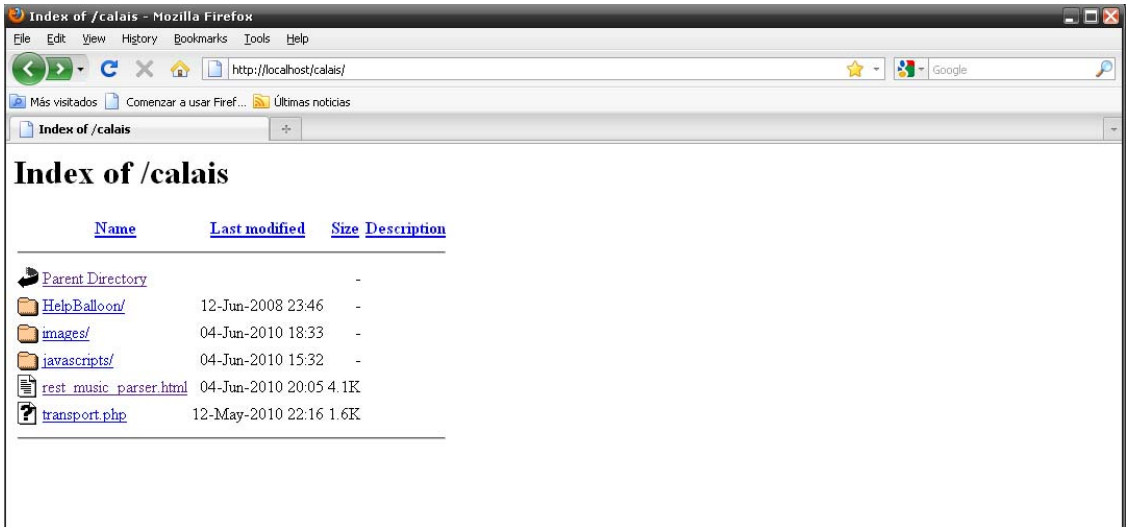


Ilustración 91: Puesta en marcha. Contenido de la solución

Donde aparecen todos los archivos y directorios de la solución. Sólo nos interesa el rest_music_parser.html, con lo que lo pulsamos (se puede acceder directamente a él a través de la url: http://localhost/calais/rest_music_parser.html), para acceder a la página principal.

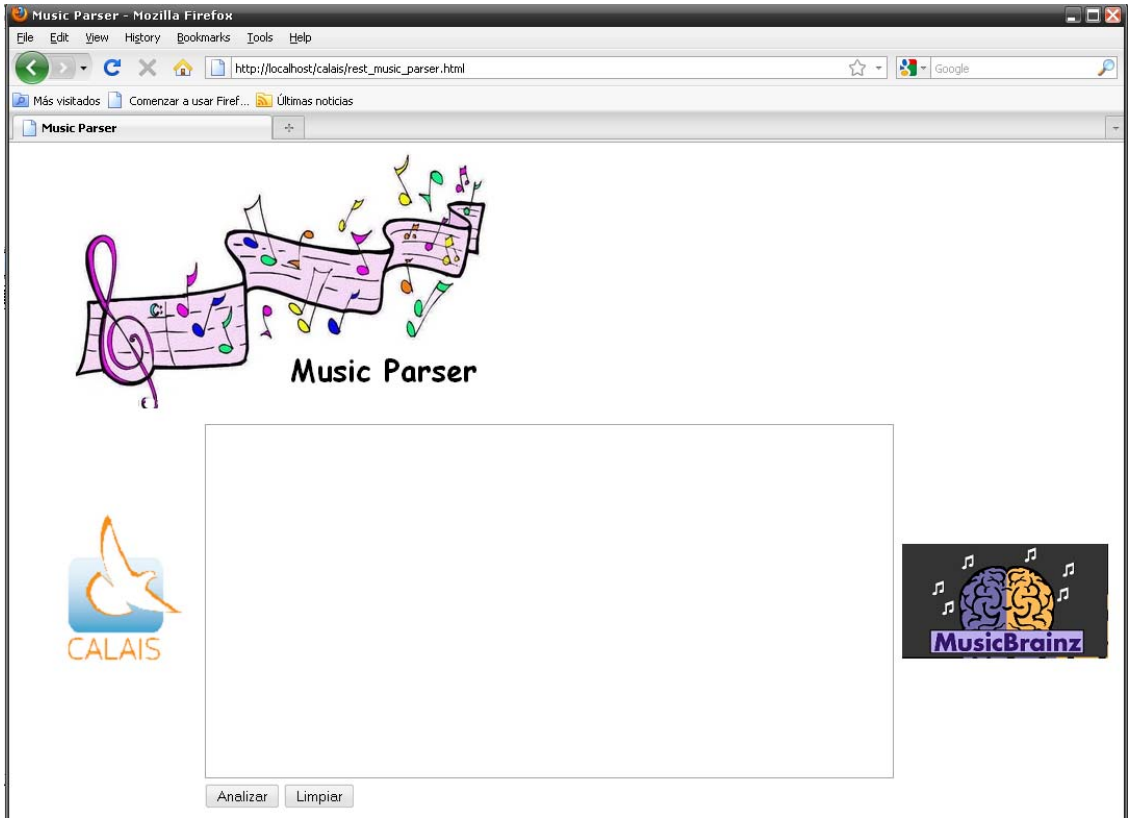


Ilustración 92: Puesta en marcha. Iniciar página

5. Introducimos en el área de texto el texto que deseamos analizar y pulsamos el botón Analizar:

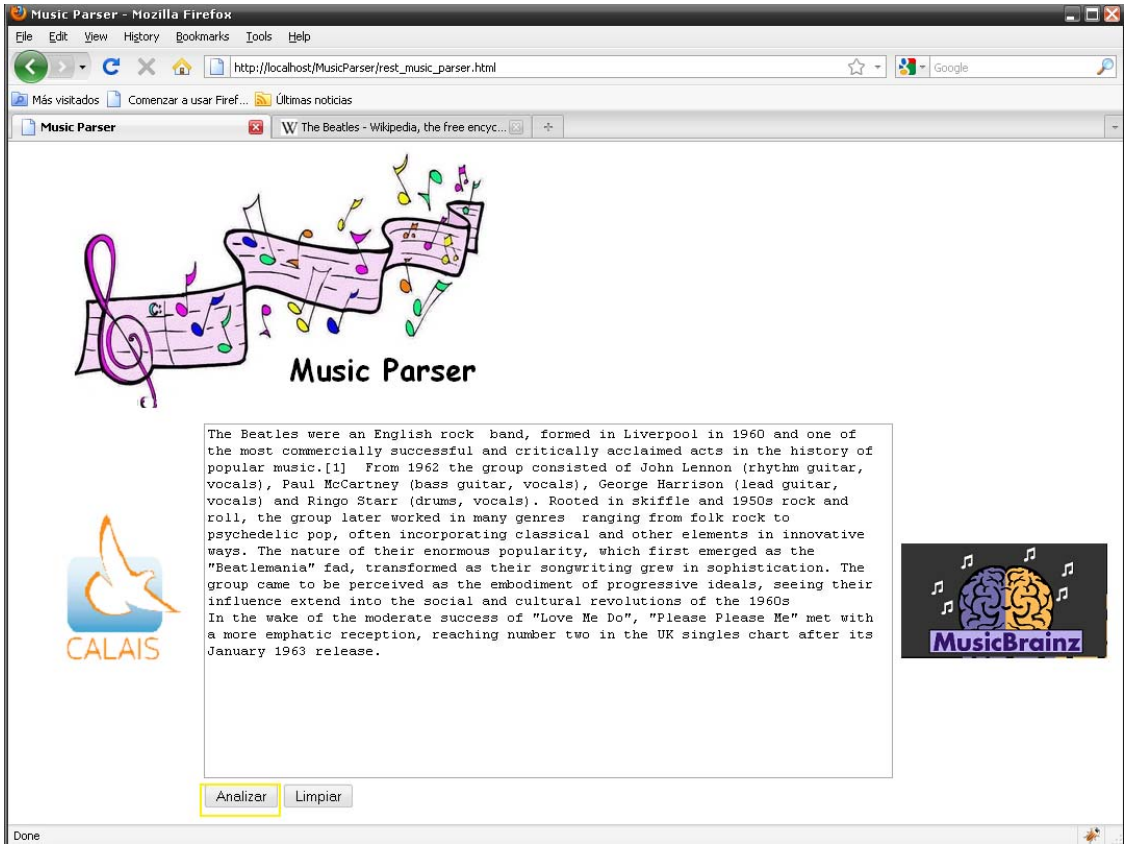


Ilustración 93: Puesta en marcha. Pulsar Analizar

Ahora se puede ver el texto procesado, es decir qué palabras ha conseguido reconocer:

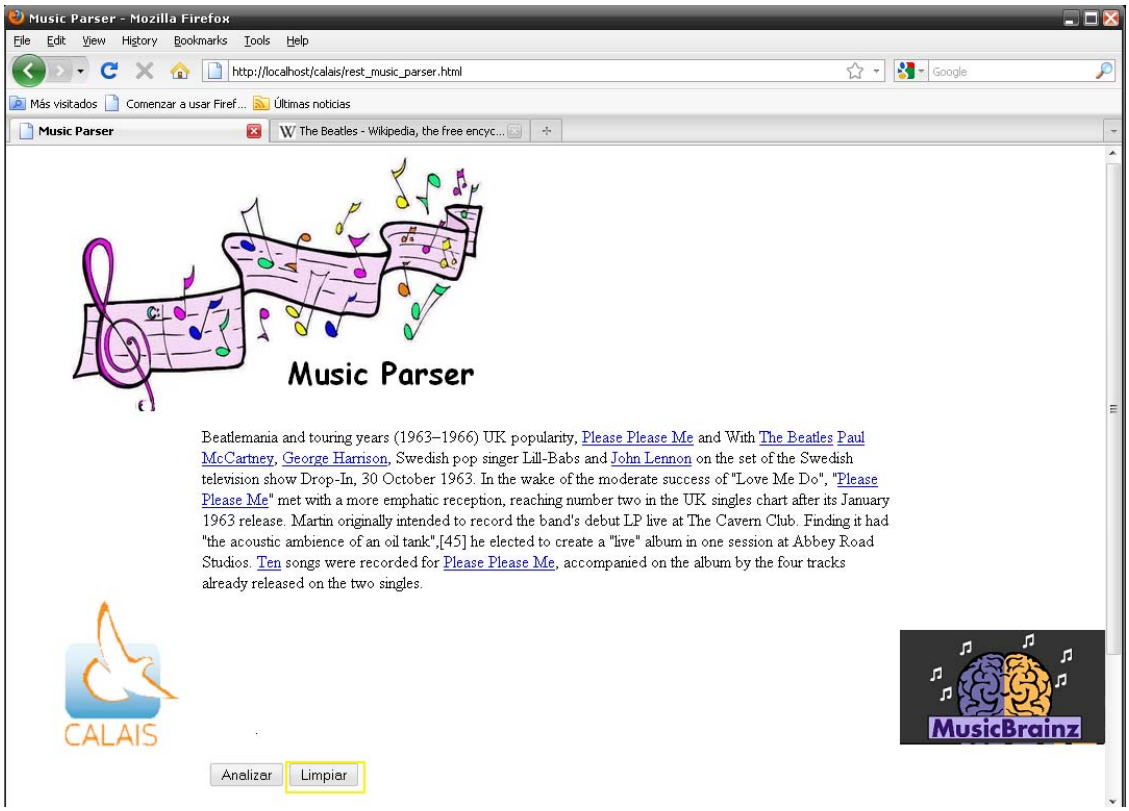


Ilustración 94: Puesta en marcha. Pulsar Limpiar

Si pulsamos el botón Limpiar volveremos a la página inicial.

Si pulsamos en cualquiera de los términos podremos obtener información adicional.

6. Para Grupos Musicales, pulsando The Beatles:

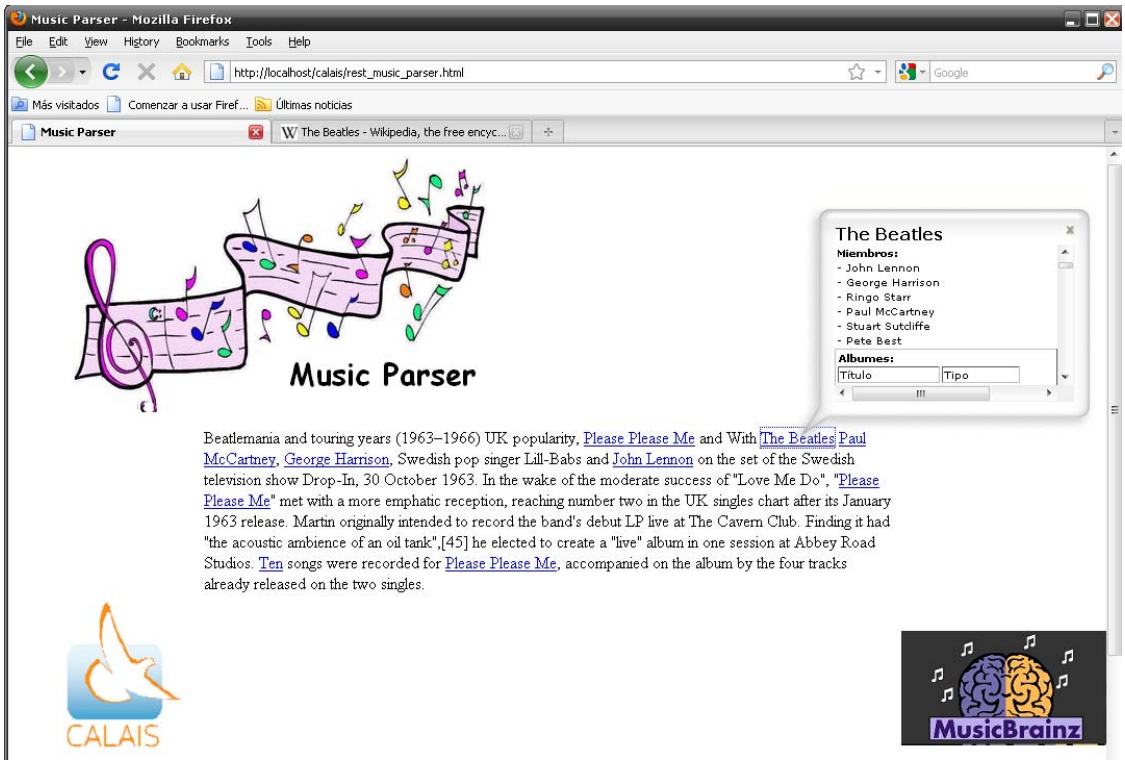


Ilustración 95: Puesta en marcha. Pulsar sobre la entidad Grupo

Por quién está compuesto el grupo, sus principales álbumes, la valoración que los usuarios de MusicBrainz han hecho del mismo y una serie de vínculos a otras Webs que permitan ampliar la información sobre el mismo.

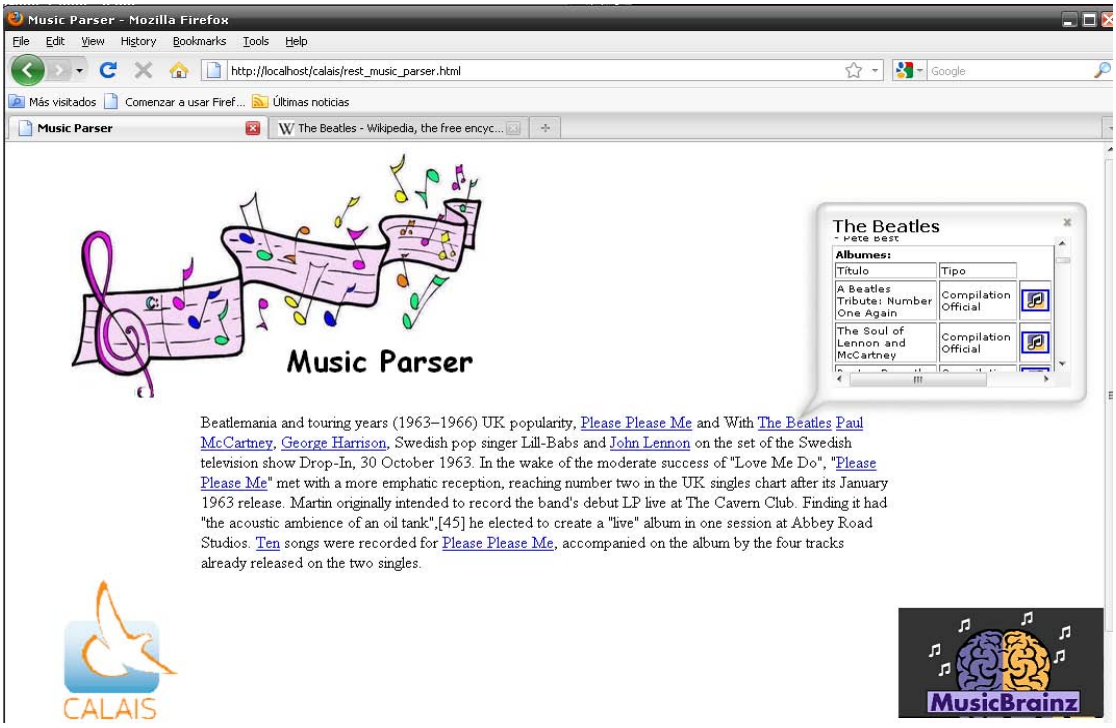


Ilustración 96: Puesta en marcha. Pulsar sobre la entidad Grupo

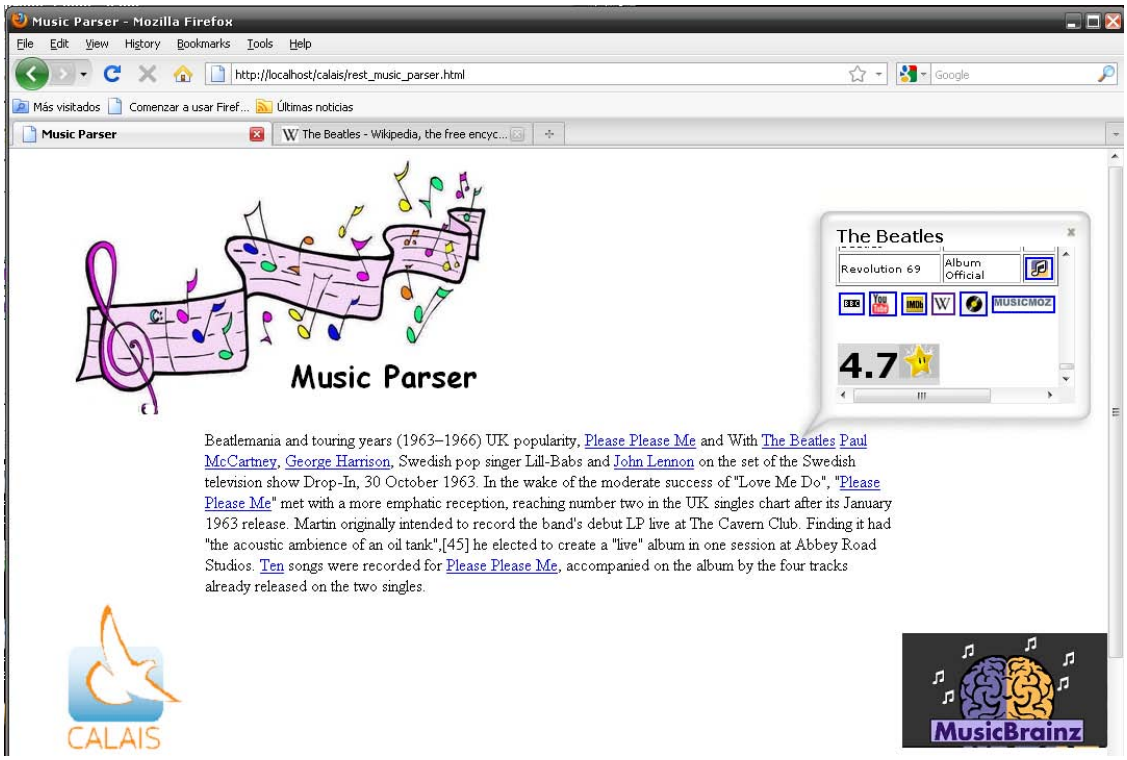


Ilustración 97: Puesta en marcha. Pulsar sobre la entidad Grupo

7. Para Artistas, pulsando John Lenon:

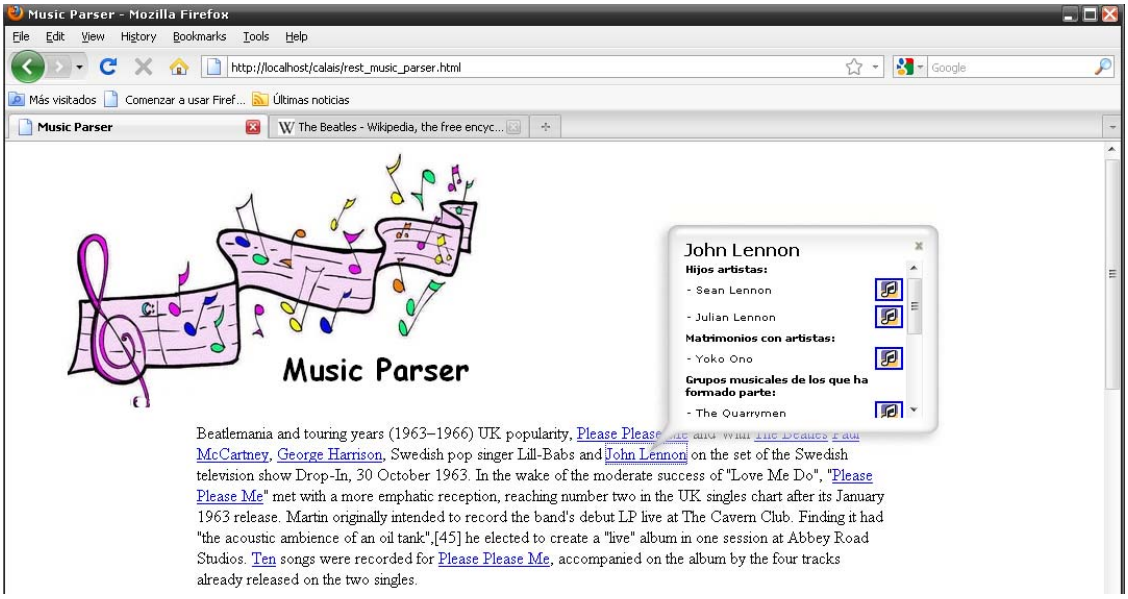


Ilustración 98: Puesta en marcha. Pulsar sobre la entidad Artista

Se muestra si tiene hijos artistas o ha estado casado con algún artista, además permite enlazar con la página de los mismos en MusicBrainz. También aparece la valoración que los usuarios de MusicBrainz han hecho del mismo y una serie de enlaces a otras Webs con más información relacionada con el artista en cuestión.

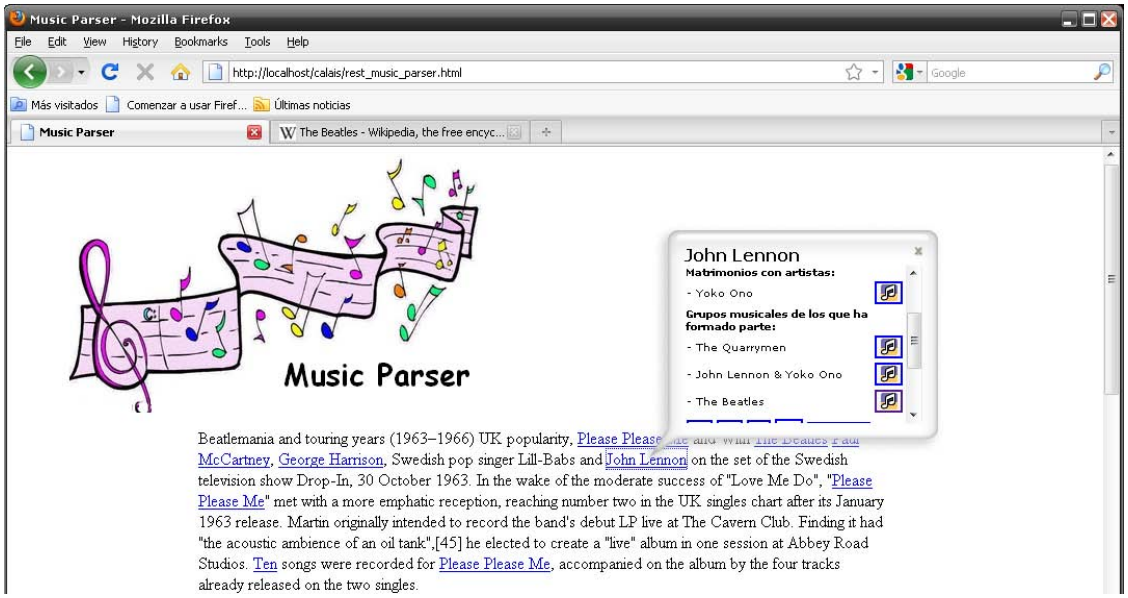


Ilustración 99: Pulsar sobre la entidad Artista

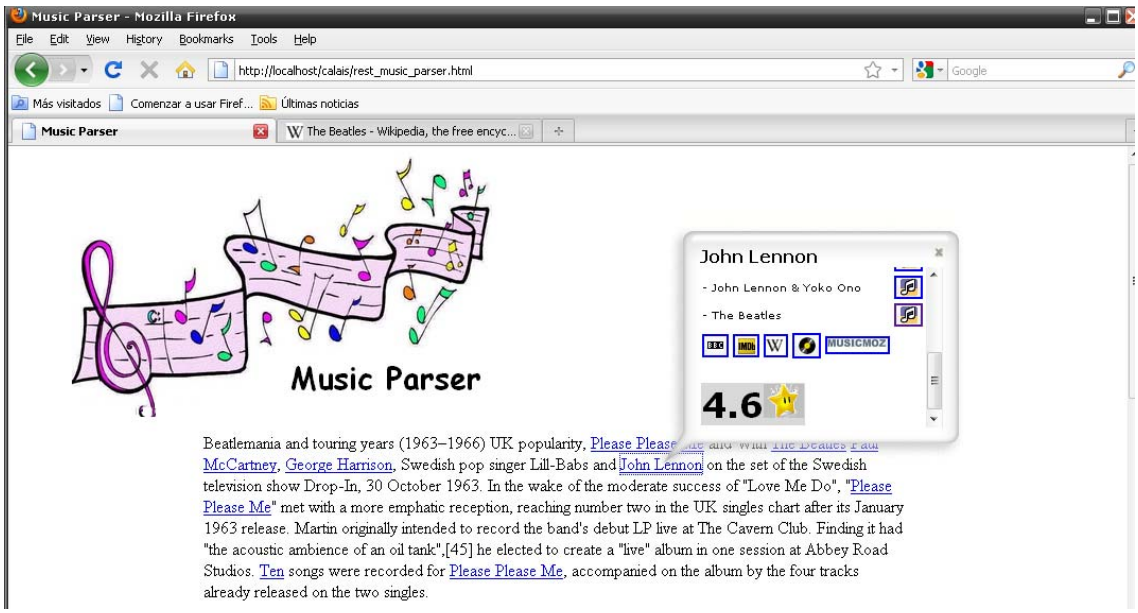


Ilustración 100: Puesta en marcha. Pulsar sobre la entidad Artista

8. Para Álbumes musicales, pulsando Please, Please Me

Ilustración 101: Puesta en marcha. Pulsar sobre la entidad Álbum

Obtenemos primero de qué grupo o artista es el álbum, luego aparece información sobre las pistas que aparecen en el álbum, su duración y su enlace a MusicBrainz.

También se muestra la valoración que los usuarios de MusicBrainz hacen del mismo y las versiones que han salido al mercado con sus formatos correspondientes.

Ilustración 102: Puesta en marcha. Pulsar sobre la entidad Álbum

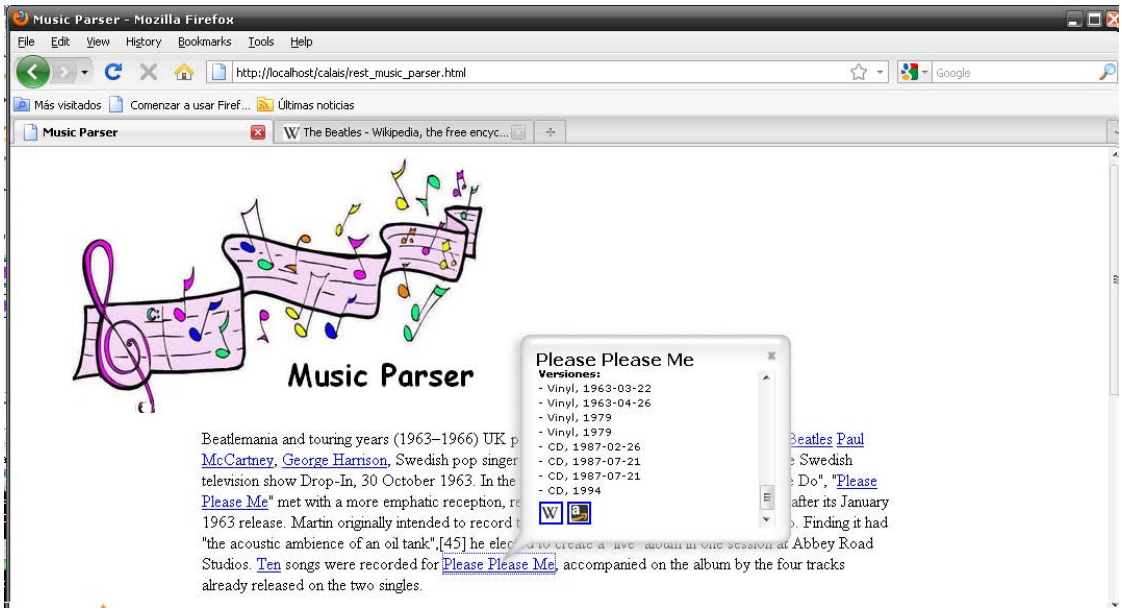
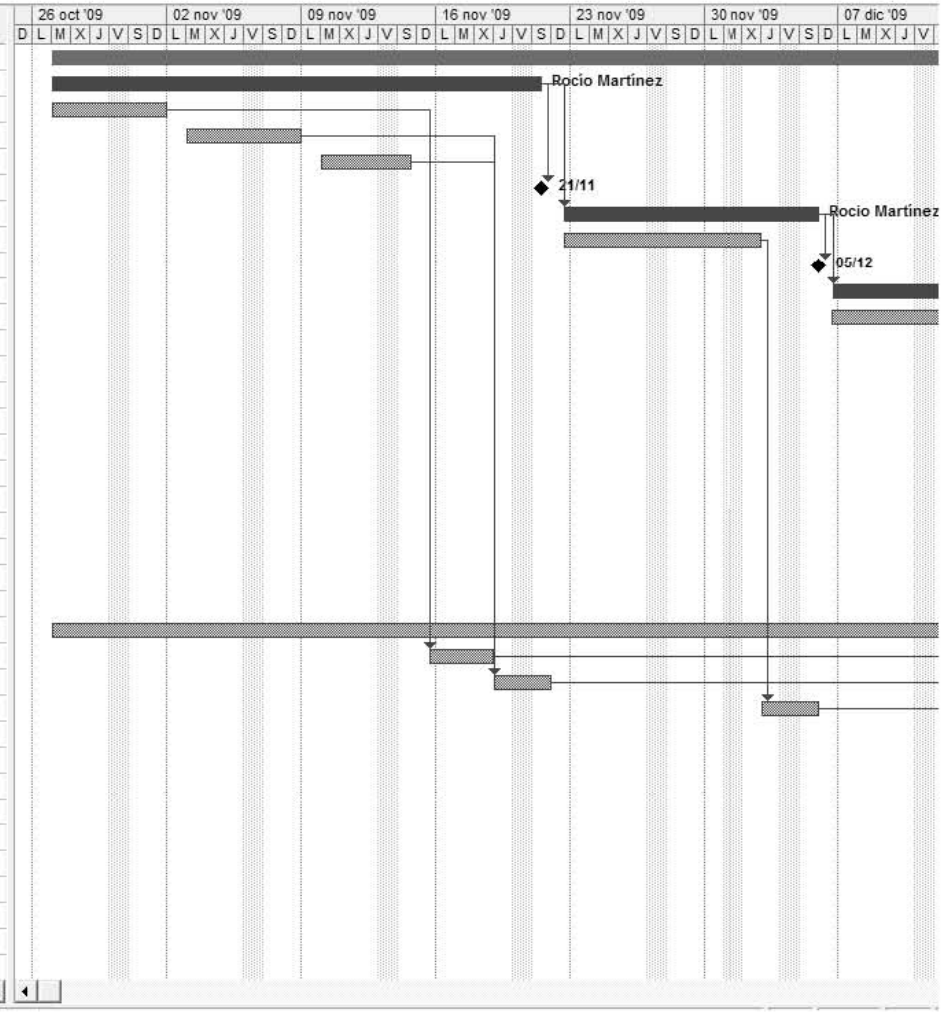


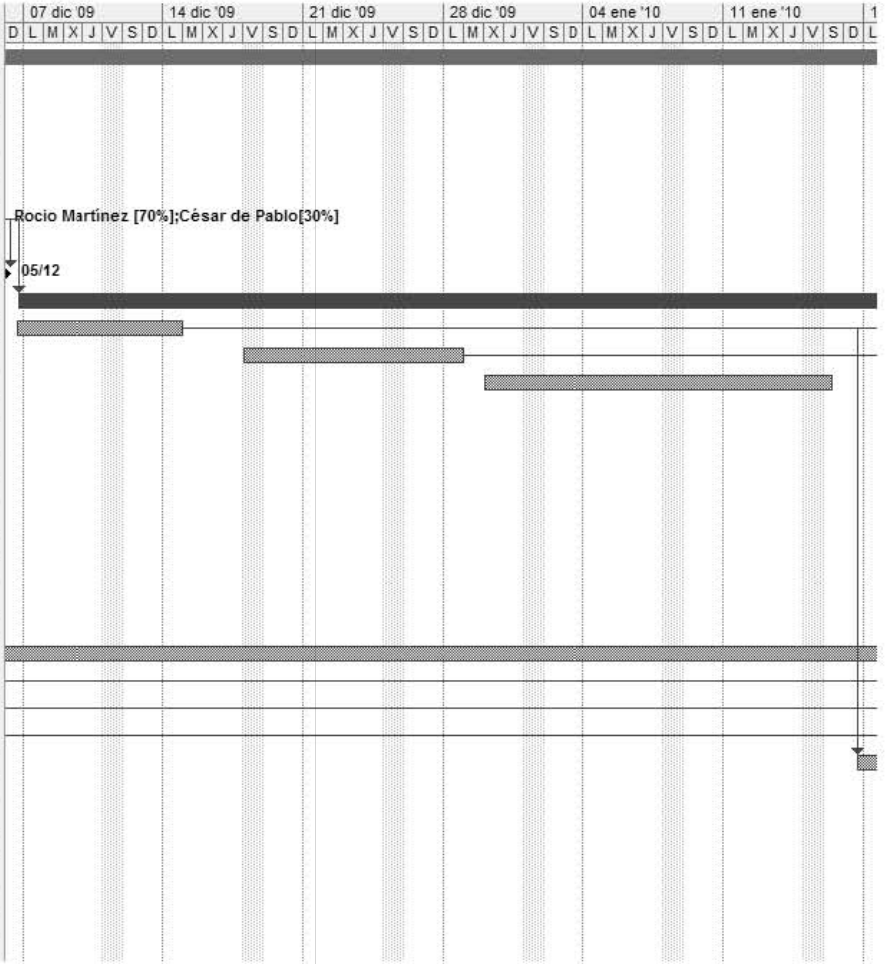
Ilustración 103: Puesta en marcha. Pulsar sobre la entidad Álbum

7.2. Anexo II: Planificación

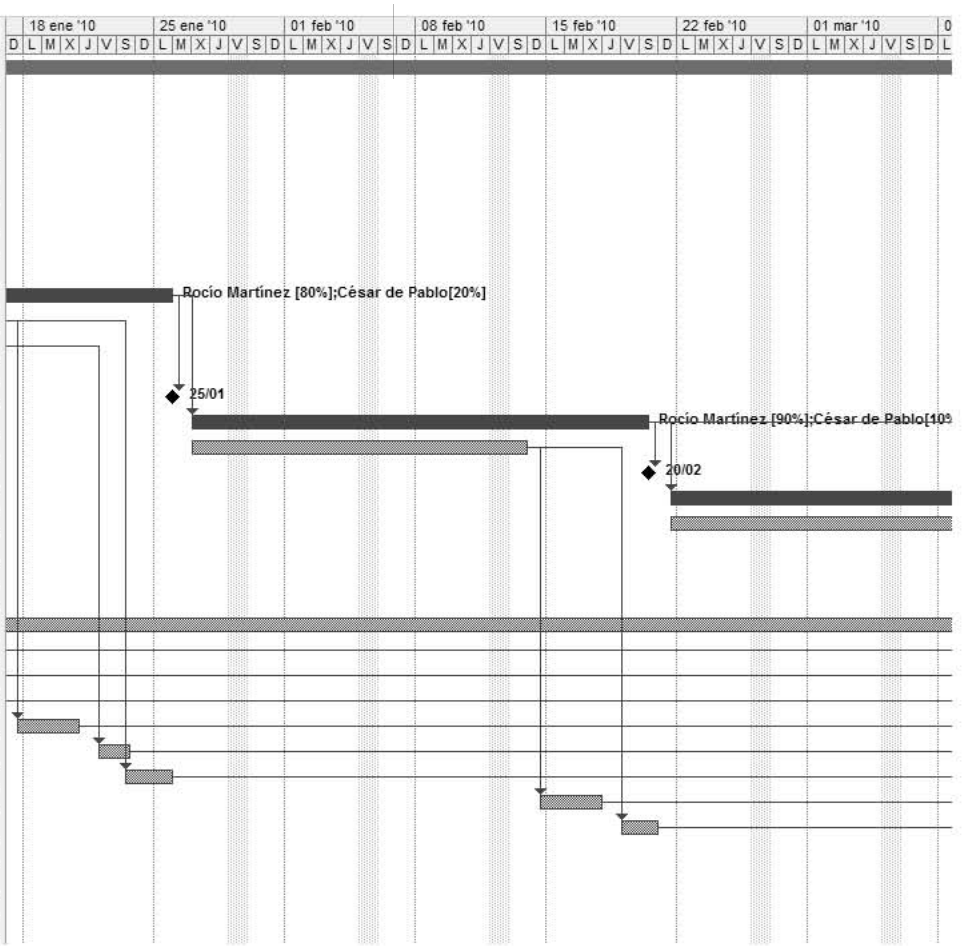
	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
1	**PFC**	**75,13 días**	**lun 26/10/09**	**mar 06/07/10**	
2	**T1 Documentación de las herramientas**	**7,5 días**	**lun 26/10/09**	**sáb 21/11/09**	
3	T1.1 Web Semántica	2 días	lun 26/10/09	dom 01/11/09	
4	T1.2 Open Calais	2 días	lun 02/11/09	dom 08/11/09	
5	T1.3 MusicBrainz	1,5 días	lun 09/11/09	sáb 14/11/09	
6	Reunión	0 días	sáb 21/11/09	sáb 21/11/09	2
7	**T2 Toma de requisitos**	**3,75 días**	**dom 22/11/09**	**sáb 05/12/09**	**2**
8	T2.1 Toma de requisitos	2,75 días	dom 22/11/09	mié 02/12/09	
9	Reunión	0 días	sáb 05/12/09	sáb 05/12/09	7
10	**T3 Análisis del Sistema**	**15 días**	**dom 06/12/09**	**lun 25/01/10**	**7**
11	T3.1 Análisis del sistema	2,5 días	dom 06/12/09	lun 14/12/09	
12	T3.2 Arquitectura del sistema	3,5 días	jue 17/12/09	lun 28/12/09	
13	T3.3 Formación Ajax, JSON y PHP	5 días	mar 29/12/09	sáb 16/01/10	
14	Reunión	0 días	lun 25/01/10	lun 25/01/10	10
15	**T4 Diseño del Sistema**	**7,5 días**	**mar 26/01/10**	**sáb 20/02/10**	**10**
16	T4.1 Diseño del sistema	5,5 días	mar 26/01/10	sáb 13/02/10	
17	Reunión	0 días	sáb 20/02/10	sáb 20/02/10	15
18	**T5 Construcción del Sistema**	**26,25 días**	**dom 21/02/10**	**sáb 22/05/10**	**15**
19	T5.1 Desarrollo del código	17 días	dom 21/02/10	lun 19/04/10	
20	T5.2 Pruebas	7 días	mar 20/04/10	jue 13/05/10	
21	Reunión	0 días	sáb 22/05/10	sáb 22/05/10	15
22	**T6 Memoria final**	**75,13 días**	**sáb 22/05/10**	**dom 30/01/11**	**18**
23	T6.1 Construcción de la memoria	75 días	lun 26/10/09	mar 06/07/10	
24	*D1.1 Documento Web Semántica*	*1 día*	*dom 15/11/09*	*mié 18/11/09*	*3*
25	*D1.2 Documenro Servicios Web*	*1 día*	*mié 18/11/09*	*sáb 21/11/09*	*4;5*
26	*D2.1 Documento Objetivos*	*1 día*	*mié 02/12/09*	*sáb 05/12/09*	*8*
27	*D3.1 Documento Tecnología utilizada*	*1 día*	*dom 17/01/10*	*mié 20/01/10*	*11*
28	*D3.2 Documento Arquitectura*	*1 día*	*jue 21/01/10*	*sáb 23/01/10*	*12*
29	*D3.3 Documento Diagramas de secuencia*	*1 día*	*sáb 23/01/10*	*lun 25/01/10*	*11*
30	*D4.1 Documento Descomposición por componentes*	*1 día*	*dom 14/02/10*	*mié 17/02/10*	*16*
31	*D.4.2 Documento OpenCalais y MusicBrainz WebServices*	*1 día*	*jue 18/02/10*	*sáb 20/02/10*	*16*
32	*D5.1 Documento de Pruebas*	*1 día*	*sáb 15/05/10*	*lun 17/05/10*	*19*
33	*D5.2 Manual de usuario*	1 día	mar 18/05/10	sáb 22/05/10	19;20
34	*D.6.1 Documento Memoria*	*13 días*	*lun 24/05/10*	*jue 08/07/10*	*24;25;26;27;28;29;30;31;32;33*
35	Reunión	0 días	jue 08/07/10	jue 08/07/10	18



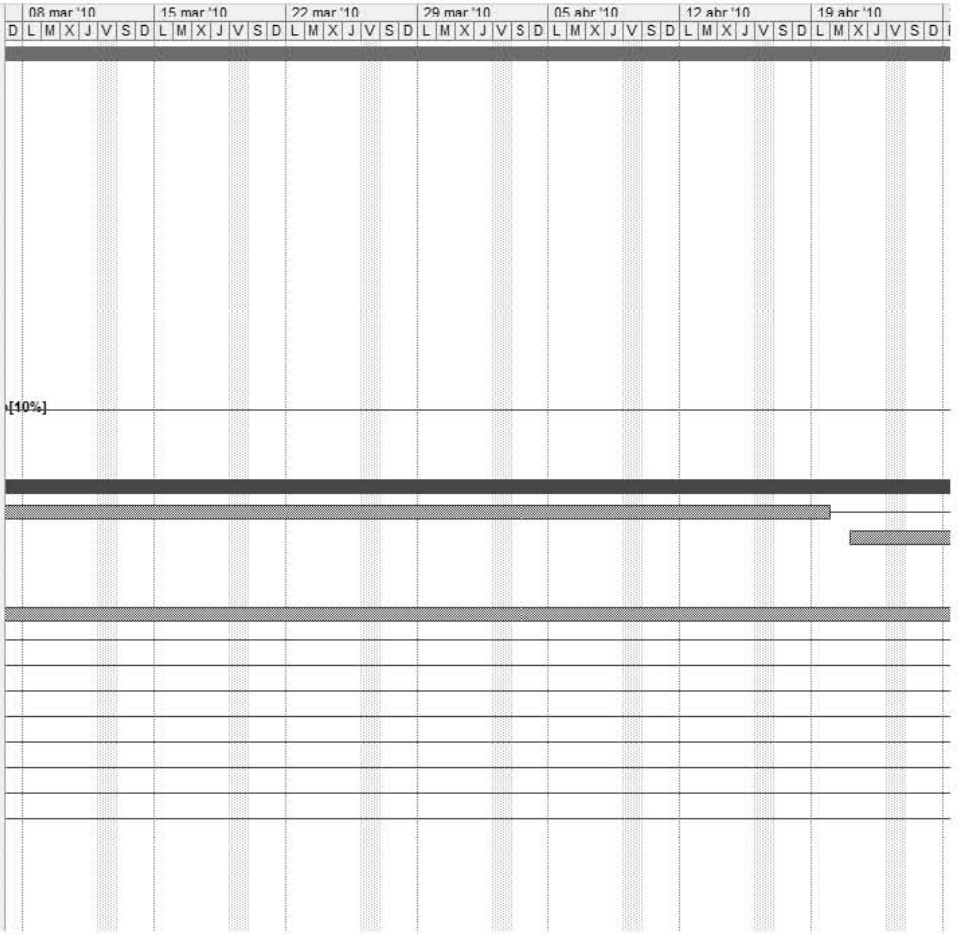
	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
1	**PFC**	**75,13 días**	**lun 26/10/09**	**mar 06/07/10**	
2	**T1 Documentación de las herramientas**	**7,5 días**	**lun 26/10/09**	**sáb 21/11/09**	
3	T1.1 Web Semántica	2 días	lun 26/10/09	dom 01/11/09	
4	T1.2 Open Calais	2 días	lun 02/11/09	dom 08/11/09	
5	T1.3 MusicBrainz	1,5 días	lun 09/11/09	sáb 14/11/09	
6	Reunión	0 días	sáb 21/11/09	sáb 21/11/09	2
7	**T2 Toma de requisitos**	**3,75 días**	**dom 22/11/09**	**sáb 05/12/09**	**2**
8	T2.1 Toma de requisitos	2,75 días	dom 22/11/09	mié 02/12/09	
9	Reunión	0 días	sáb 05/12/09	sáb 05/12/09	7
10	**T3 Análisis del Sistema**	**15 días**	**dom 06/12/09**	**lun 25/01/10**	**7**
11	T3.1 Análisis del sistema	2,5 días	dom 06/12/09	lun 14/12/09	
12	T3.2 Arquitectura del sistema	3,5 días	jue 17/12/09	lun 28/12/09	
13	T3.3 Formación Ajax, JSON y PHP	5 días	mar 29/12/09	sáb 16/01/10	
14	Reunión	0 días	lun 25/01/10	lun 25/01/10	10
15	**T4 Diseño del Sistema**	**7,5 días**	**mar 26/01/10**	**sáb 20/02/10**	**10**
16	T4.1 Diseño del sstema	5,5 días	mar 26/01/10	sáb 13/02/10	
17	Reunión	0 días	sáb 20/02/10	sáb 20/02/10	15
18	**T5 Construcción del Sistema**	**26,25 días**	**dom 21/02/10**	**sáb 22/05/10**	**15**
19	T5.1 Desarrollo del código	17 días	dom 21/02/10	lun 19/04/10	
20	T5.2 Pruebas	7 días	mar 20/04/10	jue 13/05/10	
21	Reunión	0 días	sáb 22/05/10	sáb 22/05/10	15
22	**T6 Memoria final**	**75,13 días**	**sáb 22/05/10**	**dom 30/01/11**	**18**
23	T6.1 Construcción de la memoria	75 días	lun 26/10/09	mar 06/07/10	
24	*D1.1 Documento Web Semántica*	*1 día*	*dom 15/11/09*	*mié 18/11/09*	*3*
25	*D1.2 Documenro Servicios Web*	*1 día*	*mié 18/11/09*	*sáb 21/11/09*	*4;5*
26	*D2.1 Documento Objetivos*	*1 día*	*mié 02/12/09*	*sáb 05/12/09*	*8*
27	*D3.1 Documento Tecnología utilizada*	*1 día*	*dom 17/01/10*	*mié 20/01/10*	*11*
28	*D3.2 Documento Arquitectura*	*1 día*	*jue 21/01/10*	*sáb 23/01/10*	*12*
29	*D3.3 Documento Diagramas de secuencia*	*1 día*	*sáb 23/01/10*	*lun 25/01/10*	*11*
30	*D4.1 Documento Descomposición por componentes*	*1 día*	*dom 14/02/10*	*mié 17/02/10*	*16*
31	*D.4.2 Documentc OpenCalais y MusicBrainz WebServices*	*1 día*	*jue 18/02/10*	*sáb 20/02/10*	*16*
32	*D5.1 Documento de Pruebas*	*1 día*	*sáb 15/05/10*	*lun 17/05/10*	*19*
33	*D5.2 Manual de usuario*	1 día	mar 18/05/10	sáb 22/05/10	19;20
34	*D.6.1 Documentc Memoria*	*13 días*	*lun 24/05/10*	*jue 08/07/10*	*24;25;26;27;28;29;30;31;32;33*



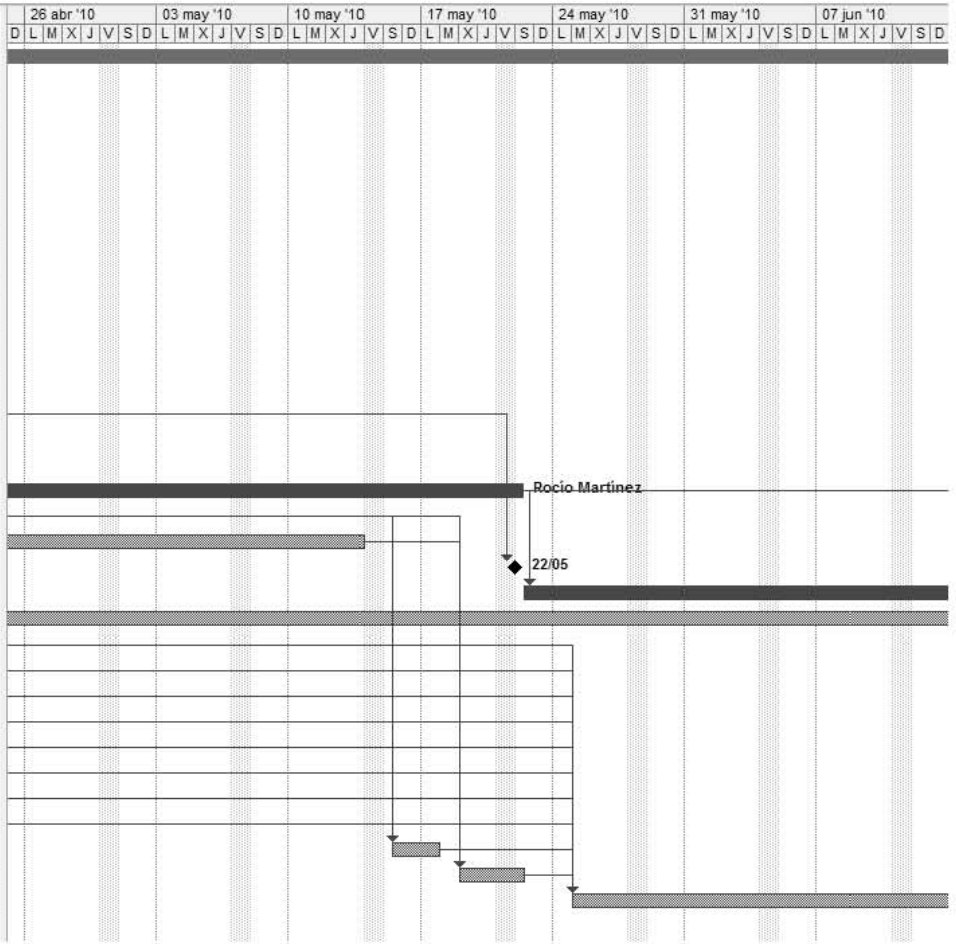
	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
1	**PFC**	**75,13 días**	**lun 26/10/09**	**mar 06/07/10**	
2	**T1 Documentación de las herramientas**	**7,5 días**	**lun 26/10/09**	**sáb 21/11/09**	
3	T1.1 Web Semántica	2 días	lun 26/10/09	dom 01/11/09	
4	T1.2 Open Calais	2 días	lun 02/11/09	dom 08/11/09	
5	T1.3 MusicBrainz	1,5 días	lun 09/11/09	sáb 14/11/09	
6	Reunión	0 días	sáb 21/11/09	sáb 21/11/09	2
7	**T2 Toma de requisitos**	**3,75 días**	**dom 22/11/09**	**sáb 05/12/09**	**2**
8	T2.1 Toma de requisitos	2,75 días	dom 22/11/09	mié 02/12/09	
9	Reunión	0 días	sáb 05/12/09	sáb 05/12/09	7
10	**T3 Análisis del Sistema**	**15 días**	**dom 06/12/09**	**lun 25/01/10**	**7**
11	T3.1 Análisis del sistema	2,5 días	dom 06/12/09	lun 14/12/09	
12	T3.2 Arquitectura del sistema	3,5 días	jue 17/12/09	lun 28/12/09	
13	T3.3 Formación Ajax, JSON y PHP	5 días	mar 29/12/09	sáb 16/01/10	
14	Reunión	0 días	lun 25/01/10	lun 25/01/10	10
15	**T4 Diseño del Sistema**	**7,5 días**	**mar 26/01/10**	**sáb 20/02/10**	**10**
16	T4.1 Diseño del sistema	5,5 días	mar 26/01/10	sáb 13/02/10	
17	Reunión	0 días	sáb 20/02/10	sáb 20/02/10	15
18	**T5 Construcción del Sistema**	**26,25 días**	**dom 21/02/10**	**sáb 22/05/10**	**15**
19	T5.1 Desarrollo del código	17 días	dom 21/02/10	lun 19/04/10	
20	T5.2 Pruebas	7 días	mar 20/04/10	jue 13/05/10	
21	Reunión	0 días	sáb 22/05/10	sáb 22/05/10	15
22	**T6 Memoria final**	**75,13 días**	**sáb 22/05/10**	**dom 30/01/11**	**18**
23	T6.1 Construcción de la memoria	75 días	lun 26/10/09	mar 06/07/10	
24	*D1.1 Documento Web Semántica*	*1 día*	*dom 15/11/09*	*mié 18/11/09*	*3*
25	*D1.2 Documenro Servicios Web*	*1 día*	*mié 18/11/09*	*sáb 21/11/09*	*4;5*
26	*D2.1 Documento Objetivos*	*1 día*	*mié 02/12/09*	*sáb 05/12/09*	*8*
27	*D3.1 Documento Tecnología utilizada*	*1 día*	*dom 17/01/10*	*mié 20/01/10*	*11*
28	*D3.2 Documento Arquitectura*	*1 día*	*jue 21/01/10*	*sáb 23/01/10*	*12*
29	*D3.3 Documento Diagramas de secuencia*	*1 día*	*sáb 23/01/10*	*lun 25/01/10*	*11*
30	*D4.1 Documento Descomposición por componentes*	*1 día*	*dom 14/02/10*	*mié 17/02/10*	*16*
31	*D.4.2 Documento OpenCalais y MusicBrainz WebServices*	*1 día*	*jue 18/02/10*	*sáb 20/02/10*	*16*
32	*D5.1 Documento de Pruebas*	*1 día*	*sáb 15/05/10*	*lun 17/05/10*	*19*
33	*D5.2 Manual de usuario*	1 día	mar 18/05/10	sáb 22/05/10	19;20
34	*D.6.1 Documento Memoria*	*13 días*	*lun 24/05/10*	*jue 08/07/10*	*24;25;26;27;28;29;30;31;32;33*
35	Reunión	0 días	jue 08/07/10	jue 08/07/10	18



	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
1	**PFC**	**75,13 días**	**lun 26/10/09**	**mar 06/07/10**	
2	**T1 Documentación de las herramientas**	**7,5 días**	**lun 26/10/09**	**sáb 21/11/09**	
3	T1.1 Web Semántica	2 días	lun 26/10/09	dom 01/11/09	
4	T1.2 Open Calais	2 días	lun 02/11/09	dom 08/11/09	
5	T1.3 MusicBrainz	1,5 días	lun 09/11/09	sáb 14/11/09	
6	Reunión	0 días	sáb 21/11/09	sáb 21/11/09	2
7	**T2 Toma de requisitos**	**3,75 días**	**dom 22/11/09**	**sáb 05/12/09**	**2**
8	T2.1 Toma de requisitos	2,75 días	dom 22/11/09	mié 02/12/09	
9	Reunión	0 días	sáb 05/12/09	sáb 05/12/09	7
10	**T3 Análisis del Sistema**	**15 días**	**dom 06/12/09**	**lun 25/01/10**	**7**
11	T3.1 Análisis del sistema	2,5 días	dom 06/12/09	lun 14/12/09	
12	T3.2 Arquitectura del sistema	3,5 días	jue 17/12/09	lun 28/12/09	
13	T3.3 Formación Ajax, JSON y PHP	5 días	mar 29/12/09	sáb 16/01/10	
14	Reunión	0 días	lun 25/01/10	lun 25/01/10	10
15	**T4 Diseño del Sistema**	**7,5 días**	**mar 26/01/10**	**sáb 20/02/10**	**10**
16	T4.1 Diseño del sistema	5,5 días	mar 26/01/10	sáb 13/02/10	
17	Reunión	0 días	sáb 20/02/10	sáb 20/02/10	15
18	**T5 Construcción del Sistema**	**26,25 días**	**dom 21/02/10**	**sáb 22/05/10**	**15**
19	T5.1 Desarrollo del código	17 días	dom 21/02/10	lun 19/04/10	
20	T5.2 Pruebas	7 días	mar 20/04/10	jue 13/05/10	
21	Reunión	0 días	sáb 22/05/10	sáb 22/05/10	15
22	**T6 Memoria final**	**75,13 días**	**sáb 22/05/10**	**dom 30/01/11**	**18**
23	T6.1 Construcción de la memoria	75 días	lun 26/10/09	mar 06/07/10	
24	*D1.1 Documento Web Semántica*	*1 día*	*dom 15/11/09*	*mié 18/11/09*	*3*
25	*D1.2 Documenro Servicios Web*	*1 día*	*mié 18/11/09*	*sáb 21/11/09*	*4;5*
26	*D2.1 Documento Objetivos*	*1 día*	*mié 02/12/09*	*sáb 05/12/09*	*8*
27	*D3.1 Documento Tecnología utilizada*	*1 día*	*dom 17/01/10*	*mié 20/01/10*	*11*
28	*D3.2 Documento Arquitectura*	*1 día*	*jue 21/01/10*	*sáb 23/01/10*	*12*
29	*D3.3 Documento Diagramas de secuencia*	*1 día*	*sáb 23/01/10*	*lun 25/01/10*	*11*
30	*D4.1 Documento Descomposición por componentes*	*1 día*	*dom 14/02/10*	*mié 17/02/10*	*16*
31	*D.4.2 Documento OpenCalais y MusicBrainz WebServices*	*1 día*	*jue 18/02/10*	*sáb 20/02/10*	*16*
32	*D5.1 Documento de Pruebas*	*1 día*	*sáb 15/05/10*	*lun 17/05/10*	*19*
33	*D5.2 Manual de usuario*	1 día	mar 18/05/10	sáb 22/05/10	19;20
34	*D.6.1 Documento Memoria*	*13 días*	*lun 24/05/10*	*jue 08/07/10*	*24;25;26;27;28;29;30;31;32;33*
35	Reunión	0 días	jue 08/07/10	jue 08/07/10	18



	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
1	**PFC**	**75,13 días**	**lun 26/10/09**	**mar 06/07/10**	
2	**T1 Documentación de las herramientas**	**7,5 días**	**lun 26/10/09**	**sáb 21/11/09**	
3	T1.1 Web Semántica	2 días	lun 26/10/09	dom 01/11/09	
4	T1.2 Open Calais	2 días	lun 02/11/09	dom 08/11/09	
5	T1.3 MusicBrainz	1,5 días	lun 09/11/09	sáb 14/11/09	
6	Reunión	0 días	sáb 21/11/09	sáb 21/11/09	2
7	**T2 Toma de requisitos**	**3,75 días**	**dom 22/11/09**	**sáb 05/12/09**	**2**
8	T2.1 Toma de requisitos	2,75 días	dom 22/11/09	mié 02/12/09	
9	Reunión	0 días	sáb 05/12/09	sáb 05/12/09	7
10	**T3 Análisis del Sistema**	**15 días**	**dom 06/12/09**	**lun 25/01/10**	**7**
11	T3.1 Análisis del sistema	2,5 días	dom 06/12/09	lun 14/12/09	
12	T3.2 Arquitectura del sistema	3,5 días	jue 17/12/09	lun 28/12/09	
13	T3.3 Formación Ajax, JSON y PHP	5 días	mar 29/12/09	sáb 16/01/10	
14	Reunión	0 días	lun 25/01/10	lun 25/01/10	10
15	**T4 Diseño del Sistema**	**7,5 días**	**mar 26/01/10**	**sáb 20/02/10**	**10**
16	T4.1 Diseño del sistema	5,5 días	mar 26/01/10	sáb 13/02/10	
17	Reunión	0 días	sáb 20/02/10	sáb 20/02/10	15
18	**T5 Construcción del Sistema**	**26,25 días**	**dom 21/02/10**	**sáb 22/05/10**	**15**
19	T5.1 Desarrollo del código	17 días	dom 21/02/10	lun 19/04/10	
20	T5.2 Pruebas	7 días	mar 20/04/10	jue 13/05/10	
21	Reunión	0 días	sáb 22/05/10	sáb 22/05/10	15
22	**T6 Memoria final**	**75,13 días**	**sáb 22/05/10**	**dom 30/01/11**	**18**
23	T6.1 Construcción de la memoria	75 días	lun 26/10/09	mar 06/07/10	
24	*D1.1 Documento Web Semántica*	*1 día*	*dom 15/11/09*	*mié 18/11/09*	*3*
25	*D1.2 Documenro Servicios Web*	*1 día*	*mié 18/11/09*	*sáb 21/11/09*	*4;5*
26	*D2.1 Documento Objetivos*	*1 día*	*mié 02/12/09*	*sáb 05/12/09*	*8*
27	*D3.1 Documento Tecnología utilizada*	*1 día*	*dom 17/01/10*	*mié 20/01/10*	*11*
28	*D3.2 Documento Arquitectura*	*1 día*	*jue 21/01/10*	*sáb 23/01/10*	*12*
29	*D3.3 Documento Diagramas de secuencia*	*1 día*	*sáb 23/01/10*	*lun 25/01/10*	*11*
30	*D4.1 Documento Descomposición por componentes*	*1 día*	*dom 14/02/10*	*mié 17/02/10*	*16*
31	*D.4.2 Documento OpenCalais y MusicBrainz WebServices*	*1 día*	*jue 18/02/10*	*sáb 20/02/10*	*16*
32	*D5.1 Documento de Pruebas*	*1 día*	*sáb 15/05/10*	*lun 17/05/10*	*19*
33	*D5.2 Manual de usuario*	1 día	mar 18/05/10	sáb 22/05/10	19;20
34	*D.6.1 Documento Memoria*	*13 días*	*lun 24/05/10*	*jue 08/07/10*	*24;25;26;27;28;29;30;31;32;33*
35	Reunión	0 días	jue 08/07/10	jue 08/07/10	18



	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
1	**PFC**	**75,13 días**	**lun 26/10/09**	**mar 06/07/10**	
2	**T1 Documentación de las herramientas**	**7,5 días**	**lun 26/10/09**	**sáb 21/11/09**	
3	T1.1 Web Semántica	2 días	lun 26/10/09	dom 01/11/09	
4	T1.2 Open Calais	2 días	lun 02/11/09	dom 08/11/09	
5	T1.3 MusicBrainz	1,5 días	lun 09/11/09	sáb 14/11/09	
6	Reunión	0 días	sáb 21/11/09	sáb 21/11/09	2
7	**T2 Toma de requisitos**	**3,75 días**	**dom 22/11/09**	**sáb 05/12/09**	**2**
8	T2.1 Toma de requisitos	2,75 días	dom 22/11/09	mié 02/12/09	
9	Reunión	0 días	sáb 05/12/09	sáb 05/12/09	7
10	**T3 Análisis del Sistema**	**15 días**	**dom 06/12/09**	**lun 25/01/10**	**7**
11	T3.1 Análisis del sistema	2,5 días	dom 06/12/09	lun 14/12/09	
12	T3.2 Arquitectura del sistema	3,5 días	jue 17/12/09	lun 28/12/09	
13	T3.3 Formación Ajax, JSON y PHP	5 días	mar 29/12/09	sáb 16/01/10	
14	Reunión	0 días	lun 25/01/10	lun 25/01/10	10
15	**T4 Diseño del Sistema**	**7,5 días**	**mar 26/01/10**	**sáb 20/02/10**	**10**
16	T4.1 Diseño del sistema	5,5 días	mar 26/01/10	sáb 13/02/10	
17	Reunión	0 días	sáb 20/02/10	sáb 20/02/10	15
18	**T5 Construcción del Sistema**	**26,25 días**	**dom 21/02/10**	**sáb 22/05/10**	**15**
19	T5.1 Desarrollo del código	17 días	dom 21/02/10	lun 19/04/10	
20	T5.2 Pruebas	7 días	mar 20/04/10	jue 13/05/10	
21	Reunión	0 días	sáb 22/05/10	sáb 22/05/10	15
22	**T6 Memoria final**	**13,56 días**	**sáb 22/05/10**	**mar 06/07/10**	**18**
23	T6.1 Construcción de la memoria	75 días	lun 26/10/09	mar 06/07/10	
24	*D1.1 Documento Web Semántica*	*1 día*	*dom 15/11/09*	*mié 18/11/09*	*3*
25	*D1.2 Documenro Servicios Web*	*1 día*	*mié 18/11/09*	*sáb 21/11/09*	*4;5*
26	*D2.1 Documento Objetivos*	*1 día*	*mié 02/12/09*	*sáb 05/12/09*	*8*
27	*D3.1 Documento Tecnología utilizada*	*1 día*	*dom 17/01/10*	*mié 20/01/10*	*11*
28	*D3.2 Documento Arquitectura*	*1 día*	*jue 21/01/10*	*sáb 23/01/10*	*12*
29	*D3.3 Documento Diagramas de secuencia*	*1 día*	*sáb 23/01/10*	*lun 25/01/10*	*11*
30	*D4.1 Documento Descomposición por componentes*	*1 día*	*dom 14/02/10*	*mié 17/02/10*	*16*
31	*D.4.2 Documento OpenCalais y MusicBrainz WebServices*	*1 día*	*jue 18/02/10*	*sáb 20/02/10*	*16*
32	*D5.1 Documento de Pruebas*	*1 día*	*sáb 15/05/10*	*lun 17/05/10*	*19*
33	*D5.2 Manual de usuario*	1 día	mar 18/05/10	sáb 22/05/10	19;20
34	*D.6.1 Documento Memoria*	*13 días*	*lun 24/05/10*	*jue 08/07/10*	*24;25;26;27;28;29;30;31;32;33*
35	Reunión	0 días	jue 08/07/10	jue 08/07/10	18

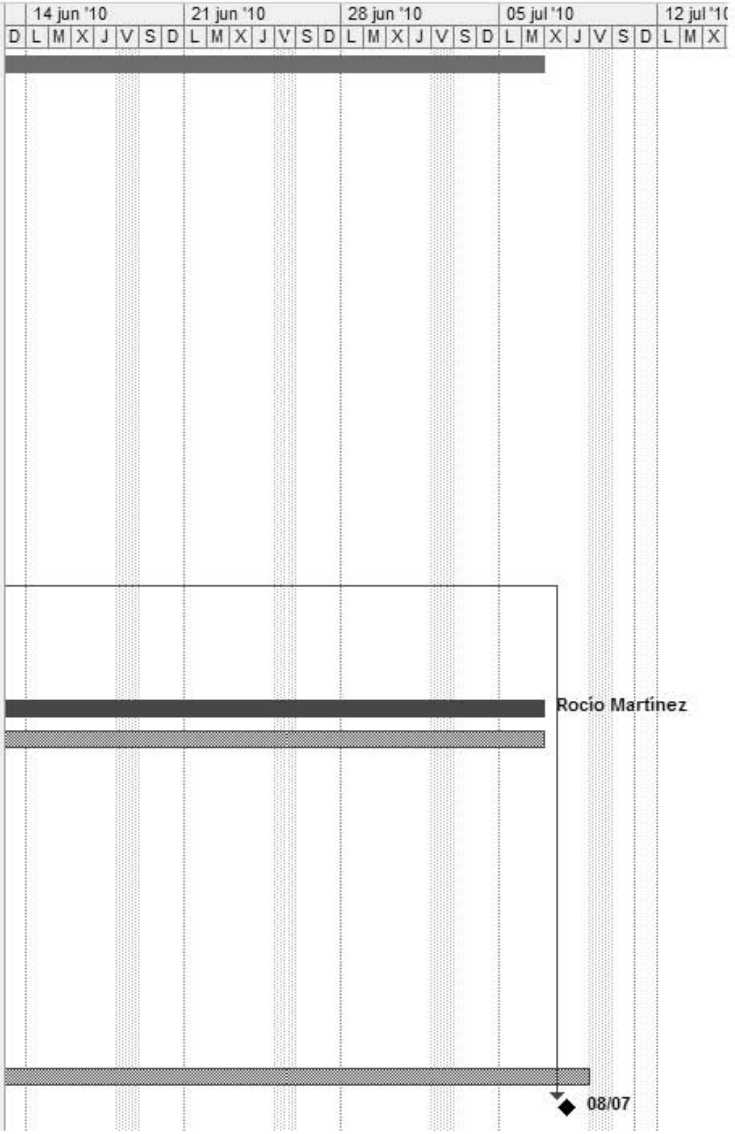


Ilustración 104: Planificación del proyecto. Diagrama de Gantt

1.1. *Anexo III: Presupuesto*

UNIVERSIDAD CARLOS III DE MADRID
Escuela Politécnica Superior

PRESUPUESTO DE PROYECTO

1.- Autor:

Rocío Martínez Vidaurrázaga

2.- Departamento:

3.- Descripción del Proyecto:

- Titulo **MusicParser**
- Duración (meses) **8**

Tasa de costes Indirectos: **20%**

4.- Presupuesto total del Proyecto (valores en Euros):

36.309,00 Euros

5.- Desglose presupuestario (costes directos)

PERSONAL

Apellidos y nombre	N.I.F. (no rellenar - solo a titulo informativo)	Categoría	Dedicación (hombres mes) [a)]	Coste hombre mes	Coste (Euro)	Firma de conformidad
Martínez Vidaurrázaga, Rocío	30970345V	Ingeniero	4,57	2.694,39	12.313,36	
de Pablo Sánchez, César		Ingeniero Senior	1	4.289,54	4.289,54	

					0,00	
					0,00	
					0,00	
			Hombres mes	**5,57**	**Total**	16.602,90

[a] 1 Hombre mes = 131,25 horas. Máximo anual de dedicación de 12 hombres mes (1575 horas)
Máximo anual para PDI de la Universidad Carlos III de Madrid de 8,8 hombres mes (1.155 horas)

EQUIPOS

Descripción	Coste (Euro)	% Uso dedicado proyecto	Dedicación (meses)	Periodo de depreciación	Coste imputable [d]
PC Intel Core 1 Duo	900,00	100	5	60	68,55
PC Intel Core 1 Duo	900,00	10	1	60	1,50
		100		60	0,00
		100		60	0,00
		100		60	0,00
					0,00
				Total	70,05

[d] Fórmula de cálculo de la Amortización:

$$\frac{A}{B} \text{xCxD}$$

A = nº de meses desde la fecha de facturación en que el equipo es utilizado
B = periodo de depreciación (60 meses)
C = coste del equipo (sin IVA)
D = % del uso que se dedica al proyecto (habitualmente 100%)

SUBCONTRATACIÓN DE TAREAS

Descripción	Empresa	Coste imputable

	Total	0,00

OTROS COSTES DIRECTOS DEL PROYECTO[e)]

Descripción	Empresa	Costes imputable
	Total	0,00

[e)] Este capítulo de gastos incluye todos los gastos no contemplados en los conceptos anteriores, por ejemplo: fungible, viajes y dietas, otros,…

6.- Resumen de costes

Presupuesto Costes Totales	**Presupuesto Costes Totales**
Personal	16.603
Amortización	70
Subcontratación de tareas	0
Costes de funcionamiento	0
Costes Indirectos	3.335
Total	20.008

Ilustración 105: Presupuesto del proyecto